Міністерство освіти і науки України

Харківський національний університет імені В.Н. Каразіна

Факультет комп'ютерних наук

Спеціальність 125 «Кібербезпека»

Освітня програма «Безпека інформаційних та комунікаційних систем»

«Допущено до захисту»
Зав.кафедрою БІСТ
Сергій РАССОМАХІН

_____

« » 2022р.

Пояснювальна записка

до кваліфікаційної роботи магістра

на тему: «Дослідження евристичних алгоритмів пошуку високонелінійних S-блоків»

оцінка « »

Голова ЕК

Доценко С.І. _____

Керівник: доцент ЗВО, к.т.н.
Полуяненко М. О.

Рецензент: професор ЗВО, д.т.н.
Кузнецов О. О.

Виконавець: студентка групи КБ-61
Арищенко А. Д.

Харків – 2022

РЕФЕРАТ

Загальний об’єм роботи становить 63 сторінки, використано 32 джерела. Дипломна робота складається зі змісту, вступу, трьох розділів, висновків, списку використаної літератури, 1 додатку, 1 таблиці та 20 рисунків.

Об’єкт дослідження – сучасні симетричні криптографічні алгоритми зокрема, блокові шифри, які є складовими шифрами, тобто являють собою композицію простих перетворень, евристичні алгоритми, властивості S-блоків, нелінійність, алгоритми пошуку високонелінійних S-блоків.

Предмет досліджень – високонелінійні S-блоки, які є основною складовою шифрування з симетричними ключами, яка виконує підстановки.

Метою роботи є: визначити роль евристичних алгоритмів пошуку високонелінійних S-блоків. Для виконання поставленої мети необхідно виконати такі завдання:

– Розглянути сутність поняття високонелінійності;

– Вивчити мету та способи підвищення нелінійності;

– Розглянути сутність поняття S-блоку;

– Вивчити методи та способи генерації S-блоків;

– Розглянути особливості пошуку та використання S-блоків;

– Вивчити сутність поняття евристичного алгоритму;

– Провести аналіз прикладів використання евристичних алгоритмів пошуку та їхню суть;

– Розглянути критерії і показники, необхідні для практики реалізації алгоритмів пошуку високонелінійних S-блоків;

– Провести аналіз найефективніших алгоритмів пошуку високонелінійних S-блоків та їхню суть.

Проблема, що вирішується полягає у тому, що параметри синтезу S-блоків впливають на подальшу роботу алгоритмів, а забезпечення достатньої

криптографічної стійкості вимагає витрат таких ресурсів, як: велика часова витрата, технічна складність у реалізації, велика вартість виконання, тощо. Тому, цю проблему необхідно розглянути та вирішити, адже від властивостей S-блоків суттєво залежить криптографічна стійкість складових шифрів, що диктує необхідність синтезу блоків, які є усталеними відносно сучасних методів криптографічного аналізу.

Методами дослідження в даній роботі є: спостереження, порівняння, моніторинг, математичні методи перетворювання інформації, методи теорії захисту інформації, прикладної криптології, математичного та імітаційного моделювання, теорії ймовірності та математичної статистики, методи розробки та верифікації програмного забезпечення.

Ключові слова: СИМЕТРИЧНІ ШИФРИ, КРИПТОЛОГІЯ, ЕВРИСТИКА, ЕВРИСТИЧНІ АЛГОРИТМИ, НЕЛІНІЙНІСТЬ, S-БЛОК, ПОШУК S-БЛОКІВ, СХОДЖЕННЯ НА ПАГОРБ, ЛОКАЛЬНИЙ ПОШУК, ІМІТАЦІЯ ВІДПАЛУ, МЕТОД ГРАДІЄНТНОГО СПУСКУ, ГЕНЕТИЧНІ АЛГОРИТМИ.

ABSTRACT

The total volume of work is 63 pages, 32 sources are used. The thesis consists of a table of contents, an introduction, three sections, conclusions, a list of reference, 1 appendix, 1 table, 20 images.

The object of research is modern symmetric cryptographic algorithms, in particular, block ciphers, which are constituent ciphers, that is, they represent a composition of simple transformations, heuristic algorithms, properties of S-boxes, nonlinearity, algorithms for finding highly nonlinear S-boxes.

The subject of research is highly nonlinear S-boxes, which are the main component of encryption with symmetric keys, which performs substitutions.

The purpose of the work is: to determine the role of heuristic algorithms for searching for highly nonlinear S-boxes. In order to fulfill the set goal, the following tasks must be completed:

– Consider the essence of the concept of highly nonlinearity;
– To study the purpose and methods of increasing nonlinearity;
– Consider the essence of the S-box concept;
– To study the methods and ways of generating S-boxes;
– Consider the peculiarities of searching and using S-boxes;
– To study the essence of the concept of heuristic algorithm;
– Analyze examples of the use of heuristic search algorithms and their essence;
– Consider the criteria and indicators necessary for the practice of implementing highly nonlinear S-box search algorithms;
– Conduct an analysis of the most effective search algorithms for highly nonlinear S-boxes and their essence.

The problem to be solved is that the synthesis parameters of S-boxes affect the further operation of the algorithms, and ensuring sufficient cryptographic

stability requires spending such resources as: high time consumption, technical complexity in implementation, high cost of implementation, etc. Therefore, this problem must be considered and solved, because the cryptographic stability of the constituent ciphers depends significantly on the properties of S-boxes, which dictates the need for the synthesis of blocks that are established in relation to modern methods of cryptographic analysis.

Research methods in this work are: observation, monitoring, comparison, mathematical methods of information transformation, methods of information protection theory, applied cryptology, mathematical and simulation modeling, probability theory and mathematical statistics, software development and verification methods.

Keywords: SYMMETRICAL CIPHERS, CRYPTOLOGY, HEURISTICS, HEURISTIC ALGORITHMS, NONLINEARITY, S-BOX, S-BOXES SEARCH, HILL CLIMBING, LOCAL SEARCH, SIMULATED ANNEALING, GRADIENT DESCENT METHOD, GENETIC ALGORITHMS.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ, І ТЕРМІНІВ

FIR	–	Finite impulse response
NFIR	–	Nonlinear finite impulse response
EBT	–	Euler–Bernoulli beam theory
DES	–	Data Encryption Standard
AES	–	Advanced Encryption Standard
SPN	–	Service Principal Name
SAC	–	Strict Avalanche Criterion
PC	–	Propagation criterion
WHS	–	Walsh Hadamard spectrum
БФ	–	Булева функція

ВСТУП

Симетрична криптографія зазвичай використовується для надання послуг, пов’язаних з інформаційною безпекою, таких як конфіденційність, працездатність, простота, тощо. Симетричний криптографічний простір набув широкого застосування з високою продуктивністю та низькою складністю досягнення. З точки зору стабільності, швидкості та надійності, симетричні криптографічні перетворення є як особливою перевагою, так і простотою використання. Оскільки у більшості сучасних блокових симетричних шифрів для введення циклових ключів у алгоритм шифрування використовується лінійна операція (побітове додавання за модулем 2), S-блоки ідентифікуються за основними компонентами, які визначають нелінійні зміни та стабільність для більшості криптографічних атак.

Важливу роль у забезпеченні безпеки симетричних перетворень відіграють нелінійні випадкові вузли (булеві криптографічні функції, векторні відображення, або S-блоки). Також, функції шифрування впливають на параметр стійкості шифрів до різноманітних криптографічних атак.

Розвиток методів криптографічного аналізу призвів до розробки критеріїв S-блоку і показників продуктивності проти атак цього типу. Підвищена нелінійність S-блоку, зокрема, забезпечує захист від лінійного криптоаналізу. Тому важливо розробити швидкий метод створення S-блоків з бажаним індексом нелінійності.

В інформатиці евристичний алгоритм здатний приймати рішення між багатьма рішеннями, але він не може бути гарантовано найкращим рішенням. Зазвичай, такі алгоритми знаходять найкращі рішення, а також пошук є швидким. Такі алгоритми можуть бути точними. Тобто вони насправді знаходять найкраще рішення, але доти, доки не виявиться, що рішення насправді найкраще, вони називається евристичним.

Евристичний алгоритм виключає одну або обидві цілі; наприклад, він зазвичай знаходить дуже гарне рішення, але немає доказів того, що рішення не є насправді поганим; або він діє досить швидко, але не гарантує, що він завжди забезпечить розв’язок.

Актуальність теми даної роботи визначається необхідністю створення та удосконалення сучасних симетричних криптоалгоритмів, що підвищують надійність і захищеність інформаційно-телекомунікаційних систем передачі даних. Швидкодія та ефективність сучасних криптографічних алгоритмів в багатьох аспектах визначається властивостями використаних нелінійних перетворень, тобто S-блоків підстановки. Варто відзначити, при використанні криптографічних методів необхідно враховувати витрати на захист інформації та на реалізацію методів нападу. Є важливим досягнення компромісу між вартістю шифрування і необхідним ступенем забезпечення безпеки.

Якість S-блоків підстановки визначається вимогами критеріїв криптографічної стійкості конструкції, які засновані на певних видах атак криптоаналізу та показниками економії апаратних ресурсів, простої реалізації, швидкості роботи і т.д. Властивість протистояння атакам криптоаналізу значно покращиться зі зростанням довжини блоку. Однак зі зростанням довжини S-блока дуже ускладняються методи їх синтезу, а також довжина блоку в деяких випадках впливає на пам’ять та продуктивність. Тому необхідно віднайти баланс між ефективністю та ресурсовитратністю. Зростання довжини S-блоків підстановки позитивно позначається на їх криптографічній якості: різко зростає відстань нелінійності, падає кореляційний зв’язок виходу та входу, збільшуються ступінь алгебри нелінійності і період повернення. Однак збільшення довжини S-блоків підстановки також означає утруднення процесу їх вибору через стрімке зростання загальної кількості існуючих блоків підстановки. Зворотною стороною збільшення довжини S-блоків підстановки є стрімке збільшення пам'яті, необхідної для їх зберігання.

Отже, в даній роботі буде розглянуто поняття евристики та евристичного алгоритму, для того, щоб визначити методи та способи дослідження, розкрито суть S-блоку та висвітлено критерії пошуку та методи пошуку S-блоків. Крім того визначені головні властивості S-блоку та за результатом аналізу пояснене поняття нелінійності, як однієї з головних властивостей S-блоків. У другому розділі визначені основні алгоритми пошуку S-блоків та надано коротке пояснення щодо кожного алгоритму. У фінальній частині порівняно та проаналізовано основні алгоритми пошуку S-блоків шляхом запуску відповідних програм та аналізу результатів.

1 ТЕОРЕТИЧНІ ОСНОВИ ВИКОРИСТАННЯ ВИСОКОНЕЛІНІЙНИХ S-БЛОКІВ

1.1 Визначення поняття нелінійності в криптографії

Підстановки сучасних симетричних примітивів, які включають функції розгортання ключів, зазвичай реалізуються як таблиці перестановок. Сучасний BSSH використовує лінійну арифметику (побітове додавання по модулю 2) для подачі циклічного ключа алгоритм шифрування. S-блок – єдиний фактор, який може визначати нелінійність шифруючого перетворення та рівень стійкості до атак криптоаналізу.

Нелінійністю булевої функції визначають мінімальну відстань Хеммінга для всіх афінних функцій, що складаються з змінних n. Далі представлено зв’язок між нелінійністю булевої функції та перетворенням Уолша.

$$NL(f) = \frac{1}{2} \cdot (2^n - \max(|W(\omega)|)) \tag{1.1}$$

Нелінійність векторної булевої функції безпосередньо пов’язана з максимальним значенням таблиці відповідності як $NL(F) = 2^{n-1} - \lambda$. Зрозуміло, що ці два критерії взаємозамінні. Тому, якщо ми знаємо значення нелінійності поставлення, можна легко обчислити максимальне значення підібраної таблиці, і навпаки. Ця властивість дозволяє прискорити обчислення умови за рахунок використання найшвидшого методу обчислення однієї з умов [1].

1.1.2 Зв’язок між критерієм нелінійності та поняттям S-блоку

Відображення шифрування, зокрема шифри продукту, часто розроблені для задоволення набору вибраних критеріїв, які були встановлені формально

чи емпірично як важливі для безпеки шифру. Два основні критерії належать Шеннону, який припустив, що шифр продукту повинен бути побудований з використанням понять дифузії та плутанини. Дифузія стосується розсіювання статистичних властивостей відкритого тексту, а плутанина стосується внутрішніх операцій шифру, які створюють складні зв'язки між відкритим текстом, ключем і зашифрованим текстом. Зовсім недавно ці поняття були вдосконалені шляхом моделювання шифрів продукту та його компонентів за допомогою булевих функцій. Наприклад, якщо біт зашифрованого тексту c_i описується булевою функцією f_i, тоді загальновизнано, що кожен f_i повинен володіти комбінацією таких властивостей: збалансованість, нелінійність, невиродженість/повнота, кореляційний імунітет, задовольняти суворому лавиноподібному критерію або бути зігнутим. Ці властивості можуть спільно називатися критеріями нелінійності та можуть бути розширені кількома природними способами.

Для шифрів продукту критерії нелінійності зазвичай застосовуються до побудови S-блоків. Нелінійність шифру продукту безпосередньо залежить від вибору цих S-блоків, оскільки зазвичай S-блоки є єдиним неафінним компонентом шифру; зокрема, якщо S-блоки є афінними, тоді все відображення є афінним (як у випадку з DES).

У 1985 році Рідс і Манферделлі розробили атаку, яку вони назвали факторизацією криптосистеми. Ідея полягає в тому, що можуть існувати окремі афінні функції для відкритого тексту, зашифрованого тексту та ключа, так що в відображених доменах розмірність простору ключів була зменшена (тобто в відображеному домені певні біти ключа є виродженими). Якби це було так, тоді вартість вичерпного пошуку простору ключів була б зменшена. Для DES Рідс і Манферделлі показали, що такої факторизації круглого відображення не існує.

Узагальнюючи ці ідеї, Чаум і Евертсе визначили лінійні структури та розробили атаку на DES, яка є менш дорогою, ніж вичерпний пошук, коли DES обмежено менш ніж 8 раундами.

Криптоаналітик може мати можливість скористатися перевагами лінійних структур у f, якщо деякі з m_i у рівнянні дорівнюють нулю, таким чином усуваючи вплив деяких змінних (можливо, ключових бітів) на зашифрований текст. Меєр і Стаффельбах показали, що для навіть n бент-функції досягають максимальної відстані від класу n-розрядних функцій, які мають лінійну структуру.

Диференціальний криптоаналіз можна розглядати як розширення ідей атак, заснованих на наявності лінійних структур. Можна альтернативно стверджувати, що $b \neq 0 \in Z_2^n$ є лінійною структурою n-розрядної функції f тоді і тільки тоді, коли:

$$\sum_{X \in Z_2^n} \hat{f}(X + b) = \pm 2^n \mathrm{X} \qquad (1.3)$$

де $\hat{f}(X) = (-1^{f(X)})$. Таким чином, вхідні дані різниці b призводять до виходу різниці нуль або одиниці з імовірністю 1. У диференціальному криптоаналізі потрібно лише, щоб вхідні дані різниці ΔX приводили до відомої різниці ΔY з високою ймовірністю або з імовірністю, яка помітно перевищує значення. Евертсе визначив функцію f як таку, що має 50%-лінійну структуру відносно $b \neq 0$, якщо:

$$\sum_{X \in Z_2^n} \hat{f}(X) \cdot \hat{f}(X + b) = 0 \qquad (1.2)$$

Евертсе скептично ставився до того, що S-блоки можуть бути спроектовані, що задовольняє цю властивість для кожного $b \neq 0 \in Z_2^n, b$ для кожного вихідного біта S-блоку. Майєр і Стаффельбах пізніше визначили функцію f як ідеальну нелінійну, якщо для всіх $b \neq 0 \in Z_2^n$ є 50%-лінійною структурою для f. Якщо n-розрядна функція f є ідеальною нелінійною, тоді $\hat{f}(X) \cdot \hat{f}(X + b)$ враховуючи $b \neq 0 \in Z_2^n$ з однаковою ймовірністю спричинять вихідну різницю. Це свідчить про те, що ідеальні нелінійні лінійні функції є корисним класом функцій для побудови відображень, які є стійкими до диференціальних атак [2].

1.3 Сутність поняття S-блоку

У криптографії S-блок – основна складова симетричного, яка виконує підстановки. У блокових шифрах її частіше використовують для приховування зв'язків між ключем і шифротекстом – властивість плутанини, яка була введена Шенноном [3, с. 9-18]. Загалом, S-блок приймає m біт на вхід і перетворює їх в n біт на виході, де n не завжди дорівнює m. S-блок можна відобразити як таблицю пошуку з 2 m слів з n бітів кожне (див. Таблиця 1.1).

Таблиця 1.1 – Стан системи на вході

		Внутрішні 4 біти на вході															
		000	**001**	**010**	**011**	**100**	**101**	**110**	**111**	**000**	**001**	**010**	**011**	**100**	**101**	**110**	**111**
Зовнішні біти	**0**	010	100	100	001	111	010	011	110	000	101	011	111	101	000	110	001
	1	110	011	010	100	100	111	101	001	101	000	111	010	011	001	000	110
	0	100	010	001	011	010	101	111	000	111	001	100	101	110	011	000	110
	1	011	000	100	111	001	110	010	101	110	111	000	001	010	100	101	011

Дані 6 бітів на вході і 4-х бітів на виході знаходяться через вибір рядка, використовуючи зовнішні два біти (перший і останній), а стовпчик знаходиться по чотирьох внутрішніх бітах. Наприклад, вхід «011011» має зовнішні «01» і внутрішні біти «1101» і відповідний вихід «1001». Встановлюються, також, вимоги до проектування S-блоку:

– S-блок не є лінійною або афінною функцію від свого входу.

– Зміна 1 входового біту має як наслідок зміну щонайменше 2 бітів на виході.

– S(x) і S(x + 001100) мусять різнитись більше ніж двома бітами.

– S(x) = / = S(x+11ab00) для будь-якого вибору a і b.

– S-блоки обираються так, щоб мінімізувати різницю між кількістю 1 і 0 у будь-якому виході S-блоку за умови сталості одного входового біту.

– Зумисно відібрані S-блоки потребують для втілення значно більше термінів, ніж довільно обрані [4, с. 103].

Після винайдення диференціального криптоаналізу, Дон Копперсміт оприлюднив умови, використані при розробці S-блоків:

– Кожен S-блок повинен мати 6 біт на вході і 4 на виході (У 1974 р. це був найбільший розмір S-блоку, який можна була використати так, щоб DES вписувався в один чіп).

– Жоден виходовий біт S-блоку не повинен бути занадто близьким до лінійної функції від входових бітів. (S-блок – єдина нелінійна складова DES. В їхній нелінійності полягає сила алгоритму).

– Кожен «рядок» S-блоку повинен містити всі можливі виходи, що увипадковлює вихід.

– Якщо два входи різняться одним бітом, їх виходи мають різнитись не менше ніж двома бітами.

– Якщо два входи S-блоку різняться двома середніми бітами, їх виходи повинні різнитися щонайменше двома бітами.

– Якщо два входи S-блоку різняться своїми першими двома бітами й мають однакові останні, виходи мають бути різними.

– Для будь-якої 6-бітної різниці між входами, не більше ніж 8 з 32 пар входів, що проявляють таку різницю, можуть проявлятись в такій самій різниці виходів.

З цифровою трансформацією змінилися пріоритети індивідів, суспільств і держав. В результаті цієї зміни інформаційна безпека є проблемою, якою не

можна нехтувати [5, с. 138]. Для того, щоб алгоритм блочного шифру вважався безпечним, необхідно виконати дві основні вимоги. Ці вимоги є плутаниною та дифузією. Алгоритми блокового шифрування базуються на криптографічних компонентах, відомих як блоки підстановки (S-блоки), що забезпечують потребу в змішуванні властивостей.

Саме тому сценарії атак зазвичай зосереджені на криптографічному компоненті. Вищезазначений підхід проектування має низькі значення для критерію нелінійності з критеріїв проектування S-блоку. Алгоритми оптимізації привертають увагу як поширений підхід, який останнім часом використовується в літературі для вирішення вищезазначеної проблеми.

1.4 Основні властивості S-блоків

Безпека даних залежить від процесу підстановки. Підстановка – нелінійне перетворення, яке здійснює заплутаність бітів. Він надає криптосистемі властивість плутанини, описану Шенноном. Він припустив, що надійні шифри можна побудувати шляхом багаторазового поєднання замін із транспозицією. Найбільш ранні блочні шифри були простими мережами, які об’єднували схеми підстановки і перестановки, і називаються підстановочними мережами (SPN). У сучасному алгоритмі шифрування нелінійне перетворення є суттєвим і доведено, що воно є сильним криптографічним примітивом проти лінійного та диференціального криптоаналізу.

1.4.1 Приклад S-блоку в нелінійному алгоритмі AES

Прикладом нелінійного алгоритму перетворення є Advanced Encryption Standard (AES). Цей стандарт визначає алгоритм Rijndael. В алгоритмі Rijndael S-блок є найважливішою частиною через алгоритм шифрування. Алгоритм шифрування означає, що він вимагає, щоб ключ був такої ж довжини, як і повідомлення, яке буде закодоване.

Однак це викликає найбільшу затримку алгоритму шифрування. На Рисунку 1.1 показано нелінійні перетворення S-блоку в AES. Компонент S-блок, який використовується в AES, є фіксованим і не змінюється. Статичний S-блок використовуватиме той самий S-блок у кожному раунді, тоді як залежний від ключа або динамічний S-блок змінюватиметься в раунді S- блоку залежно від ключа та кількості раундів. Алгоритм динамічного або залежного ключа повинен бути згенерований для підвищення криптографічної стійкості системи шифрування AES. S-блоки, залежні від значень ключів, повільніші, але більш безпечні, ніж незалежні. При розробці та аналізі AES слід враховувати криптографічні властивості S-блоку, особливо лавинний ефект [6].

	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
00	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
10	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
20	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
30	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
40	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
50	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
60	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
70	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
80	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
90	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a0	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b0	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c0	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d0	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e0	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f0	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Рисунок 1.1 – S-блок в AES

AES – симетричний блочний пароль, що означає, що він використовує той же ключ для шифрування і розшифрування. Однак AES значно

відрізняється від DES багатьма способами. Алгоритм Rijndael дозволяє використовувати різноманітні розміри блоків і ключів, а не тільки 64- і 56-бітні блоки і ключі DES. Насправді, блок і ключ можуть бути обрані незалежно від 128, 160, 192, 224, 256 біт і не повинні бути однаковими. Однак стандарт AES вказує, що алгоритм може приймати тільки три варіанти ключа, 128-бітний розмір блоку і 128, 192, 256-бітний. Залежно від використовуваної версії, типова назва змінюється на AES-128, AES-192 або AES-256. Крім цих відмінностей, AES відрізняється від DES тим, що це не структура Фейстеля. Блок даних використовується для зміни іншої половини блоку даних в структурі Фейстеля, а потім половини міняються місцями. У цьому випадку весь блок даних в кожному типі обробляється паралельно протягом кожного раунду з використанням замін і перестановок.

Від довжини ключа залежить кілька параметрів AES. Наприклад, якщо розмір використовуваного ключа дорівнює 128, то число раундів становить відповідно 10 12, 192 14 і 256 біт. В даний час найпоширенішим розміром ключа є 128-бітний ключ. Тому цей опис алгоритму AES описує цю реалізацію [7]. Rijndael розрахований на:

- Опір всім відомим атакам.
- Швидкість і компактність коду в широкому діапазоні платформ.
- Легкість проектування.

Загальну структуру AES можна побачити на Рисунку 1.2.

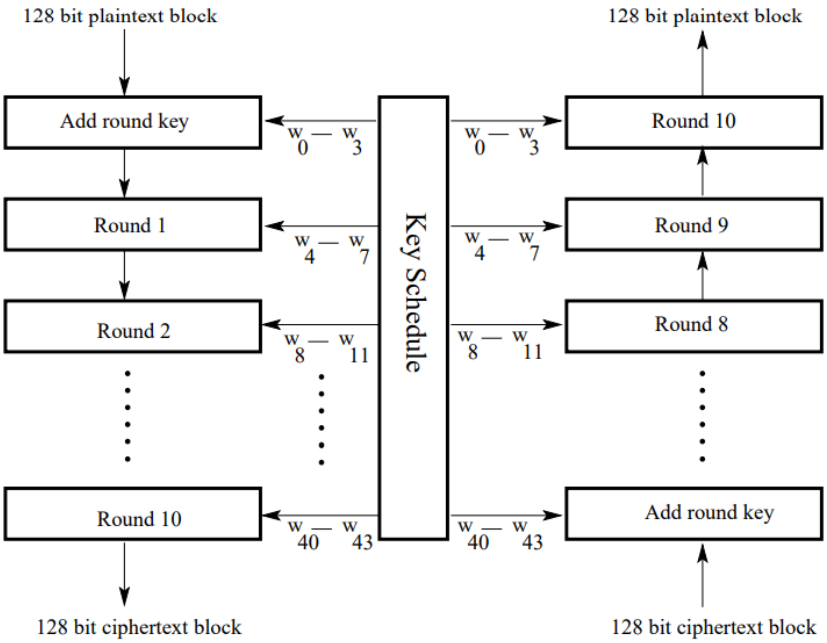

Рисунок 1.2 – Загальна структура AES для випадку 128-бітного ключа шифрування

Підстановка – нелінійне перетворення, яке виконує заплутаність бітів. Нелінійне перетворення є суттєвою для будь-якого сучасного алгоритму шифрування, і показана як сильний криптографічний примітив проти диференційованого лінійного і криптографічного аналізу. Нелінійне перетворення реалізується як таблиці пошуку (S-скриньки). S-блок з вхідними бітами p і вихідними бітами q позначається $p \rightarrow q$. DES використовує вісім 6→4 S-блоків. S-блоки призначені для розгортання програмного забезпечення на 8-розрядних процесорах. Для 32-розрядних або 64-розрядних процесорів, S-блоки з більшою кількістю вихідних бітів забезпечують високу ефективність. S-скриньки можна вибрати випадковим чином, як у Snefru, можна обчислити за допомогою хаотичної карти, або мати деяку математичну структуру над скінченним полем Галуа. S-блоки, які залежать від значень ключа, повільніші, але безпечніші, ніж незалежні від ключа.

В AES S-скринька створює два перетворення в полях Галуа GF(2) і GF(2^8). S-блок є нелінійним перетворенням, коли кожен байт стану

замінюється іншим будинком за допомогою таблиці підстановки. Перше перетворення: S-блок знаходить перетворення декількох байт в полі GF(2^8). Через те, що він є алгебраїчним виразом, атаки можуть бути здійснені в алгебраїчній складності. Звідси випливає афінне перетворення. Афінне перетворення вибирається для перетворення підбайтів у складний алгебраїчний вираз, зберігаючи при цьому нелінійні властивості. Обидва перетворення S-блоків можуть бути виражені в матричній формі [8]:

$$S' = M \cdot S^{-1} + C$$

де символ • множення і символ + додається в поле GF(2^8). Вектор S' 8×1 позначає біти в початковому після перетворень. Зворотне перетворення S-скриньки можна отримати, помноживши обидві сторони рівняння (1.6) на M^{-1}, і виконавши зворотне перетворення умови і потім зворотне GF(2^8) [8]:

$$S^{-1} = M^{-1} \cdot S' + M^{-1} \cdot C \qquad (1.4)$$

1.4.2 Основні параметри S-блоку

Властивості S-блоків широко використовувалися як основа нових стратегій шифрування, таких як нелінійність, диференціальна однорідність і суворий критерій лавини. (*x*,*y*)-S-блок: $S: \{0,1\}^x \rightarrow \{0,1\}^y$.

Він складається з n-змінних компонентних булевих функцій: $(f_1(x_1, \dots, x_n), f_2(x_1, \dots, x_n), \dots, f_n(x_1, \dots, x_n))$ кожна з яких повинна задовольняти властивості S-блоку.

Основні властивості S-блоку:

1) Надійність

Надійність визначається як здатність системи протистояти змінам, не адаптуючи її початкову стабільну конфігурацію.

Нехай $F = (f_1, f_2, \dots, f_n)$ це S-блок розміром $n \times n$, де f_i це компонентна функція відображення S-блоку: f_i: $f_i: \{0,1\}n \rightarrow \{0,1\}$.

$$R = (1 - \frac{N}{2^n})(1 - \frac{L}{2^n}) \qquad (1.5)$$

F має бути стійким до диференціального криптоаналізу.

2) Збалансованість

$S: \{0,1\}^n \to \{0,1\}^m$ збалансована, якщо $HW(f) = 2^{n-1}$. Значущість властивості збалансованості базується на вищій величині дисбалансу функції, що забезпечує високоймовірність лінійного наближення.

3) Строгий лавинний критерій (Strict Avalanche Criterion – SAC)

Зміна одного біта вхідних бітів S-блоку повинна спричинити зміну половини вихідних бітів S-блоку. Важче виконати аналіз зашифрованого тексту, намагаючись придумати атаку. Кажуть, що криптографічна функція, яка задовольняє наведену вище умову, задовольняє строгі критерії лавини.

4) Нелінійність

$S: \{0,1\}^x \to \{0,1\}^y$ визначається як найменше значення нелінійності всіх ненульових лінійних комбінацій x булевих функцій $f_i: \{0,1\} \to \{0,1\}, i = x - 1 \ldots, 1,0$. Нелінійність S-блоків повинна бути високою, щоб протистояти лінійному криптоаналізу.

5) Диференціальна рівномірність

Чим менша диференційна однорідність, тим краща стійкість S-блоку до диференціального криптоаналізу.

6) Лінійна апроксимація

Чим нижче значення лінійної апроксимації, тим краще стійкість S-блоку до лінійного криптоаналізу.

7) Алгебраїчна складність

Алгебраїчна складність важлива для захисту від інтерполяційних атак та інших алгебраїчних атак.

8) Фіксовані (Fp) і протилежні фіксовані точки (OFp)

Кількість цих Fp і OFp має бути якомога меншою, щоб уникнути витоку в будь-якому статистичному криптоаналізі.

9) Критерій незалежності бітів

Незалежність бітів є дуже бажаною властивістю, оскільки зі збільшенням незалежності між бітами стає важче зрозуміти та передбачити структуру системи [6].

1.4.3 Критерій строгої лавини та незалежність змінних лавини

Ідеї повноти та лавинного ефекту вперше були введені Камом і Давідою та Фейстелем відповідно. Якщо криптографічне перетворення завершено, то кожен біт зашифрованого тексту повинен залежати від усіх бітів відкритого тексту. Таким чином, якби було можливо знайти найпростіший логічний вираз для кожного біта зашифрованого тексту в термінах бітів відкритого тексту, кожен із цих виразів мав би містити всі біти відкритого тексту, якщо функція була повною. Альтернативно, якщо існує принаймні одна пара n-розрядних векторів відкритого тексту X і X_i, які відрізняються лише бітом i, а $f(X)$ і $f(X_i)$ відрізняються принаймні бітом j для всіх $\{(i,j)|1 \leq i,j \leq n\}$, то функція f має бути повною.

Щоб дане перетворення демонструвало ефект лавини, у середньому одна половина вихідних бітів повинна змінюватися кожного разу, коли один вхідний біт доповнюється. Щоб визначити, чи задана $m \times n$ (m вхідних бітів і n вихідних бітів) функція f задовольняє цю вимогу, 2^m векторів відкритого тексту потрібно розділити на 2^{m-1} пари, X і X_i, так що X і X_i відрізняються тільки в біті i. Тоді необхідно обчислити 2^{m-1} виняткових сум:

$$V_i = f(x) \oplus f(x_i) \tag{1.6}$$

Ці суми, або будемо називати лавинними векторами, кожен з яких містить n бітів, або лавинними змінними.

Якщо цю процедуру повторити для всіх i, таких що $1 \leq i \leq m$, і половина лавинних змінних дорівнює 1 для кожного i, тоді функція f має хороший лавинний ефект. Звичайно, цей метод можна застосувати, лише якщо m є досить малим; інакше кількість векторів відкритого тексту стає занадто великою. Якщо це так, то найкраще, що можна зробити, це взяти випадкову

вибірку векторів відкритого тексту X і для кожного значення i обчислити всі лавинні вектори V_i. Якщо приблизно половина отриманих лавинних змінних дорівнює 1 для всіх значень i, тоді можна зробити висновок, що функція має хороший лавинний ефект.

Поняття повноти та ефекту лавин можна об'єднати, щоб визначити нову властивість, яку називають строгим критерієм лавин. Якщо криптографічна функція має задовольняти суворий лавинний критерій, тоді кожен вихідний біт повинен змінюватися з ймовірністю половини щоразу, коли один вхідний біт доповнюється. Більш точне визначення критерію виглядає наступним чином. Розглянемо X і X_i, два n-розрядні двійкові вектори відкритого тексту, такі, що X і X_i відрізняються лише бітом i, $1 \leq i \leq n$. Нехай:

$$V_i = Y \oplus Y_i \tag{1.7}$$

де $Y = f(X), Y_i = f(X_i)$Y, а f – це криптографічне перетворення, яке розглядається. Якщо f має відповідати суворому критерію лавини, ймовірність того, що кожен біт у V_i дорівнює 1, має становити половину набору всіх можливих векторів відкритого тексту X і X_i. Це має бути вірним для всіх значень i. Знову ж таки, якщо n не є малим, дотримуватись цієї процедури для всіх можливих векторних пар X і X_i було б величезним завданням.

Альтернативним методом, який можна було б використати для визначення того, чи дане криптографічне перетворення, f, задовольняє суворий лавинний критерій, було б побудувати матрицю залежності. Спочатку генерується n-бітний випадковий вектор відкритого тексту X і отримується його відповідний m-бітний шифртекст $Y = f(X)$ (n і m будуть рівними, якщо f є оборотним перетворенням і немає розширення даних). Тоді набір із n векторів $(X_1, X_2, \dots, X_n)$ формується так, що X і X_j відрізняються лише бітом j. Потім знаходять вектори зашифрованого тексту $(Y_1, Y_2, \dots, Y_n)$, де $Y_j = f(X_j)$, і вони використовуються для отримання набору m-розрядних двійкових лавинних векторів $(V_1, V_2, \dots, V_n)$ такий, що $V_j = Y \oplus Y_j$. Ця процедура показана на Рисунку 1.3.

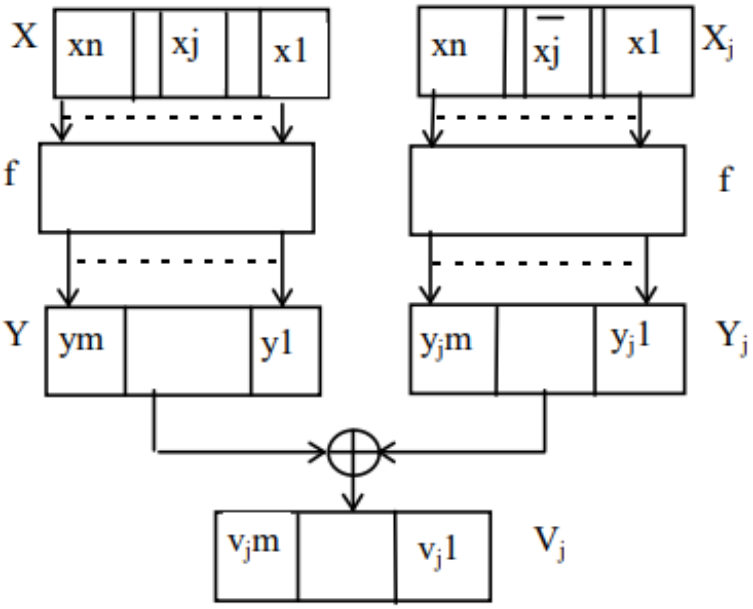

Рисунок 1.3 – Частина методу для перевірки того, чи перетворення задовольняє суворий лавинний критерій

Значення біта i у V_j (або 1, або 0) додається до елемента $a_{i,j}$ у матриці залежності $m \times n$ A. Ця процедура повторюється для великої кількості, r, випадково згенерованих векторів відкритого тексту X, і кожен елемент в A ділиться на r. Тоді кожен $a_{i,j}$ дає силу зв'язку між бітом відкритого тексту j та бітом i зашифрованого тексту. Значення 1 вказує на те, що коли біт j доповнюється у відкритому тексті, то біт зашифрованого тексту i також змінить своє значення, тоді як значення 0 вказує на те, що біт зашифрованого тексту повністю не залежить від біта відкритого тексту. Якщо всі елементи в матриці мають ненульове значення, то криптографічне перетворення завершено, і якщо воно має задовольнити суворий критерій лавини, кожен елемент повинен мати значення, близьке до половини. Отже, повнота є необхідною умовою, якщо має бути виконано суворий критерій лавини.

Друга властивість, яка здається бажаною для будь-якого криптографічного перетворення, полягає в тому, що для даного набору лавинних векторів, створених доповненням одного біта відкритого тексту, усі лавинні змінні повинні бути попарно незалежними. Щоб виміряти ступінь незалежності між парою лавинних змінних, ми можемо обчислити їхній коефіцієнт кореляції. Для двох змінних A і B:

(1.8)

$$\rho\{A,B\} = \frac{\mathrm{cov}\{A,B\}}{\sigma\{A\}\sigma\{B\}}$$

де $\rho\{A,B\}$ – коефіцієнт кореляції A і B

cov{A, B} – коваріація A і B

Для випадку двійкових змінних можна показати, що коефіцієнт кореляції 0 означає, що змінні незалежні. Крім того, змінні завжди будуть ідентичними, якщо коефіцієнт кореляції дорівнює 1, а значення -1 означає, що вони завжди будуть доповнювати одна одну.

Якщо суворий лавинний критерій або вимога незалежності лавинної змінної не задовольняються, то криптоаналітик може отримати деяку інформацію про статистичні властивості функції, яку він, імовірно, міг би використати на свою користь під час атаки на систему [9].

1.4.4 Ідеальні S-блоки

Тепер, коли представлено ці два нові критерії, в роботі [9] представлено, як створити криптографічні перетворення, які задовольняють обидві умови. Ще одна додаткова умова, яка буде накладена на такі перетворення, полягає в тому, щоб вони були оборотними. Це означає, що має існувати однозначна відповідність між векторами відкритого та зашифрованого тексту. Якщо є n вхідних/вихідних бітів для заданої функції, є $(2^n)!$ можливі оборотні перетворення. Це означає, що для чотирибітної системи буде приблизно 2×10^{13} таких функцій. Таким чином, пошук буде обмежено 4×4 (чотири вхідні/чотири вихідні біти) блоками підстановки.

Початковий крок полягає в тому, щоб знайти всі потенційно оборотні функції 4×1, які задовольняють суворому критерію лавини, які будуть об’єднані по чотири за один раз, щоб створити ящики заміни 4×4. Потенційно оборотна функція повертає значення 1 для однієї половини можливих вхідних векторів і значення 0 для іншої половини. Це необхідна, але не достатня умова, якщо S-блоки, сформовані з єдиних вихідних бітових функцій, мають бути

оборотними. Було перевірено 12 870 потенційно оборотних функцій 4×1, і було виявлено, що хоча 12 618 з них були повними, лише 1368 задовольняли суворому критерію лавини.

Ці 1368 функцій можна розділити на 9 класів еквівалентності або «сімейств». Кожне сімейство закривається на такі операції:

1) Доповнення одного або кількох вхідних бітів
2) Перестановка вхідних бітів
3) Доповнення вихідного біта

Потенційна оборотність і дотримання строгого лавиноподібного критерію зберігаються в цих операціях.

Найпростіша процедура, якої слід дотримуватися при побудові блоків підстановки, полягала б у випадковому виборі потенційно оборотних одновихідних бітових функцій зі списку тих, що задовольняють строгому критерію лавини. По-перше, ці ящики заміни перевіряються, щоб побачити, чи вони оборотні. Якщо вони задовольняють вимозі, їх потім перевіряють, щоб побачити, чи, коли кожен вхідний біт доповнюється, результуючі лавинні змінні є попарно незалежними. S-блок, який відображає обидві ці властивості, буде називатися «ідеальним» блоком заміни.

Коли використовувався метод випадкового вибору окремих вихідних бітових функцій, ймовірність того, що отримані 4×4 S-блоки будуть оборотними, становила лише $1{,}2\times10^{-3}$, і лише один S-блок у $7{,}1\times10^{5}$ був ідеальним. Під час цього пошуку було відмічено сімейства одновихідних бітових функцій, які формували ідеальні S-блоки. У спробі зменшити кількість зусиль, необхідних для створення ідеальних S-блоків, сімейства, з яких були обрані функції 4×1, були фіксовані, щоб використовувати лише комбінації, які створили ідеальні S-блоки під час початкового пошуку. Це збільшило частоту появи ідеальних S-блоків приблизно в тисячу разів. Було випробувано кілька інших підходів, які включали пом’якшення одного або обох із суворого критерію лавини та вимоги незалежності змінної лавини, але жоден не

виявився таким хорошим, як вибір єдиних вихідних бітових функцій із фіксованих сімейних комбінацій.

У процесі побудови цих S-блоків було виявлено, що якщо ящик повний або навіть ідеальний, його зворотна функція може бути неповною. Це може стати важливим, якщо ці зворотні функції використовуються в процесі дешифрування, оскільки було б бажано, щоб будь-які зміни в зашифрованому тексті впливали на всі біти у відкритому тексті випадковим чином, особливо якщо в вихідному відкритому тексті немає великої надлишковості. Повні криптографічні перетворення з оберненими, які є завершеними, описуються як двосторонні, а якщо зворотне не є повним, перетворення вважається лише одностороннім [9].

1.5 Методи генерації S-блоків

Три основні класи методів для генерації S-боксів і булевих функцій з бажаною нелінійністю:

1) Випадковий пошук
2) Методи побудови
3) Еволюційний (або генетичний) пошук найпростішого методу – випадковий пошук.

У найзагальніших термінах структура S-блоку є нелінійною функцією, яка відображає вхідний сигнал довжини m на вихідний сигнал довжини n. Ця нелінійна структура спрямована на запобігання успіху диференціальних атак у криптографічних конструкціях. Техніка проектування, запропонована Найбергом у AES S-блоку, спрямована на те, щоб найкращим чином подолати диференціальні атаки. Ще однією перевагою структури AES S-блоку є ефективне програмне забезпечення [10, с. 227].

Оскільки використовується критерій нелінійності, який є ще одним вимірюванням, що використовується для демонстрації складності структури S-блоку, то для розробленого Найбергом, значення XOR дорівнює 4, а

значення нелінійності дорівнює 112. Однак, детермінована природа математичного методу почала загрожувати безпеці системи шифрування.

Хоча методи, засновані на випадковому відборі, є більш стійкими до аналізу побічних каналів, найуспішніші результати, отримані серед досліджень, були розраховані як 10 для значення XOR і 106,75 для значення нелінійності. Коли ці значення порівнюють із показниками S-блок AES, проблема стає більш очевидною.

Випадковий пошук – це група чисельних методів оптимізації, які не вимагають збільшення градієнта для вирішення завдань оптимізації. Таким чином, випадковий пошук може бути використаний для функцій, які не є безперервними або диференційованими. Такі методи оптимізації також містять методи прямого пошуку, методи чорного ящика та недиференціальними методами.

Випадковий пошук працює шляхом постійного переміщення між оптимальними позиціями в просторі пошуку. Найкраще положення вибирається з гіперсфери з центром у поточному положенні.

Існують різні випадкові методи пошуку, які шукають з усього простору пошуку (наприклад, повний випадковий пошук або рівномірний глобальний випадковий пошук) [11].

Одним з методів, який можна використовувати для покращення продуктивності проектування на основі випадкового вибору, є алгоритми оптимізації. Однак, у цих конструкціях навантаження на обробку через алгоритми оптимізації також є недоліком.

Метод, спрямований на збільшення значення критерію нелінійності без шкоди для простоти та швидкості дозволяє зміщувати структури рядка, стовпця та обох елементів S-блоку. Показники продуктивності можна покращити шляхом зміни позицій комірок S-блоку.

Алгоритми еволюційного пошуку використовуються для вирішення задач оптимізації та моделювання шляхом послідовного вибору, об’єднання та зміни параметрів з використанням прийомів (механізмів), що нагадують

біологічну еволюцію. Ключова ідея генетичних алгоритмів – імітувати природний випадок. Тут процес природного відбору та поведінка природних систем дозволяють популяціям адаптуватися до навколишнього середовища. Популяції створюються таким чином, що особини з найвищими значеннями пристосованості з найбільшою ймовірністю відтворюються, а непридатні особини задовольняють порогові значення, встановлені шляхом багаторазового застосування набору ймовірнісних генетичних операторів, що відкидаються на основі [12, с. 368].

2 АНАЛІЗ ОСОБЛИВОСТЕЙ ВИКОРИСТАННЯ ЕВРИСТИЧНИХ АЛГОРИТМІВ

Математична оптимізація – це вибір відповідних елементів за деякими критеріями з багатьох можливих альтернатив.

У більш загальному підході завдання оптимізації полягає у максимізації або мінімізації фактичної функції шляхом систематичного вибору вхідних значень із прийнятного набору та обчислення значень функції. Узагальнення теорії інших формулювань та методів оптимізації становить велику область прикладної математики. У більш загальному сенсі оптимізація включає пошук «найкращого доступного» значення цільової функції в заданій області (або вхідних даних), що містить різні типи цільових функцій і різні типи включених областей.

2.1 Сутність поняття евристичного алгоритму

Екземпляр задачі комбінаторної оптимізації визначається набором варіантів розв’язання (тобто простором пошуку Ω) та цільовою функцією. Для вирішення такого завдання (технічно кажучи, екземпляра завдання) потрібно знайти рішення, яке оптимізує (тобто мінімізує) ту чи іншу цільову функцію. Насправді деякі комбінаторні завдання вважаються обчислювально складними на вирішення. За складністю ці завдання, зазвичай, є NP-складні. Точні алгоритми вимагають надмірного часу обчислень та експоненційно великих вхідних даних. За останні 50 років нерозв'язні обчислювальні завдання були у центрі уваги інтенсивних досліджень точних алгоритмів. На практиці алгоритми та теорія складності, математична теорія оптимізації, дослідження операцій та штучним інтелектом. Було досягнуто значного прогресу, і в даний час існує три основні підходи до таких проблем.

– Дослідження класу екземплярів, які допускають ефективні точні алгоритми.

– Розробка алгоритмів апроксимації, які працюють за поліноміальний час.

– Розробка евристичних, метаевристичних та ймовірнісних алгоритмів.

Щодо першого підходу, важливо зазначити, що експоненційний алгоритм для найгіршого випадку насправді дуже ефективний. Класичний приклад – симплексний алгоритм. Це експоненційний показник для найгіршого випадку, але зазвичай використовується для вирішення лінійного програмування за поліноміальний час. Інші алгоритми, що базуються на неявному перерахуванні (наприклад, branch-and-bound, або branch-and-cut), можуть бути дуже ефективними для практичних задач помірного розміру, хоча експоненціально для найгіршого випадку.

Точні алгоритми іноді вирішують великі нерозв’язні завдання поліноміальний час. Однак часто може відбуватися і зворотне: точні алгоритми можуть вимагати непомірно часу для малих екземплярів розміру. Це також стосується проблем розфарбування графа. В даний час немає точного алгоритму, здатного оптимально розфарбовувати випадкові графи з більш ніж 80 вершинами.

У деяких вдалих випадках алгоритми поліноміальної апроксимації можуть використовуватися для апроксимації складних задач. Це не точно, але гарантовано дає рішення коректної якості, наприклад, у межах певного коефіцієнта відхилення від оптимального рішення. Є деякі NP-повні завдання, для яких були доведені наближені алгоритми, але більшість із них не приймає точних або наближених поліноміальних алгоритмів. Насправді, більшість важкорозв’язних завдань із реального життя не можуть дати рішення за поліноміальний час із гарантовано високою якістю.

Практичним підходом до багатьох нерозв’язних задач великого розміру є евристичні алгоритми пошуку – або просто евристика. Ці алгоритми можуть

використовувати розумні ресурси та створювати прийнятні рішення, але без будь-якої теоретичної гарантії. Евристика може дати конкурентоспроможні результати не лише для добре відомої NP-повної задачі, але й для обчислювальних задач, де точні алгоритми займають дуже багато часу. Наприклад, навіть поліноміальний час виконання може бути надто довгим, щоб бути практичним у певних областях із тисячами змінних (таких як проектування електронної схеми). На практиці ми не завжди повинні знаходити найкраще рішення, лише «достатньо хороше» рішення. Крім того, оптимальне рішення, що забезпечується складними та точними алгоритмами, часто може бути досягнуте швидше за допомогою базової метаевристики.

З огляду на обчислювальну задачу, евристика – це, по суті, алгоритм «пошуку», який шукає найкраще рішення в просторі всіх потенційних рішень проблеми, просторі пошуку (званому ω), який також називають простором задачі. Враховуючи те, що неможливо перерахувати всі потенційні рішення великої нерозв’язної проблеми, питання полягає в тому, як вирішити, які частини ω досліджувати/тестувати за обмежений проміжок часу. Для цього можуть використовуватися як проблемні, так і загальні (цільові) інструкції та стратегії пошуку [13-14].

2.2 Існуючі евристичні підходи

По суті, евристичні алгоритми належать до трьох основних підходів до вирішення [18]:

1) Послідовна конструкція – наприклад Dsatur , Iterated Greedy дуже швидкі методи, але не особливо ефективні;

2) Локальний пошук; пошук табу; імітація відпалу; змінний простір пошуку або пошук по змінному сусідству; повторний локальний пошук

3) Еволюційні популяційні гібридні або розподілені методи

2.2.1 Алгоритми локального пошуку

Локальний пошук — неповний спосіб знайти рішення проблеми. Він заснований на ітеративному уточненні присвоєння змінним, доки не будуть задоволені всі обмеження. Зокрема, алгоритми локального пошуку зазвичай змінюють значення змінних у присвоєннях на кожному кроці. Нове присвоєння близьке до попереднього в просторі присвоєння, звідси і назва локальний пошук.

Усі алгоритми локального пошуку використовують функцію, яка оцінює якість присвоєння, наприклад, кількість обмежень, порушених присвоєнням. Ця сума називається вартістю присвоювання. Мета локального пошуку – знайти призначення мінімальних витрат, яке є рішенням (якщо таке є).

Існує два класи алгоритмів локального пошуку. Перший з них – жадібний або нерандомізований алгоритм. Ці алгоритми продовжують змінювати поточний розподіл, завжди намагаючись зменшити його вартість (або принаймні уникнути її збільшення). Основна проблема з цими алгоритмами полягає в тому, що можуть існувати плато, області простору розподілу, де локальні переміщення не зменшують вартість. Для вирішення цієї проблеми був винайдений другий клас алгоритмів локального пошуку. Вони залишають ці плато шляхом випадкового блукання і називаються алгоритмами рандомізованого локального пошуку.

Локальний пошук починається з першого рішення та розвивається так, що єдине рішення є, по суті, кращим рішенням. Він використовує єдиний шлях пошуку рішень замість дерева пошуку. Для кожного рішення на цьому шляху оцінюється кількість ходів у розв’язанні та застосовується найкращий хід для переходу до наступного рішення. Це повторюється знову і знову, доки не закінчиться (зазвичай через те, що його закінчився час).

Основна ідея локального пошуку:

//Ініціалізація того, що є зазвичай випадковим початковим стан

//Альтернативно, може проходити у створеному людиною початковому стані

best_found ← current_state ← RandomState()

//Тепер виконується локальний пошук

loop do

if (tired of doing it) then return best_found

else

current_state ← MakeNeighbor(current_state)

if (Cost(current_state) < Cost(best_found)) then

//Зберігається найкращий знайдений результат

best_found ← current_state

Локальний пошук дуже схожий на людські планувальники. Використовується єдиний шлях пошуку та переглядаються факти, щоб знайти правильне рішення. Тому цілком природно це реалізувати. Локальний пошук часто потрібно починати з ініціалізованого рішення, тому часто необхідно попередньо заповнити фазу евристичного рішення.

Розв’язувач локального пошуку пробує кожен хід у поточному рішенні та вибирає найкращий прийнятий хід як крок.

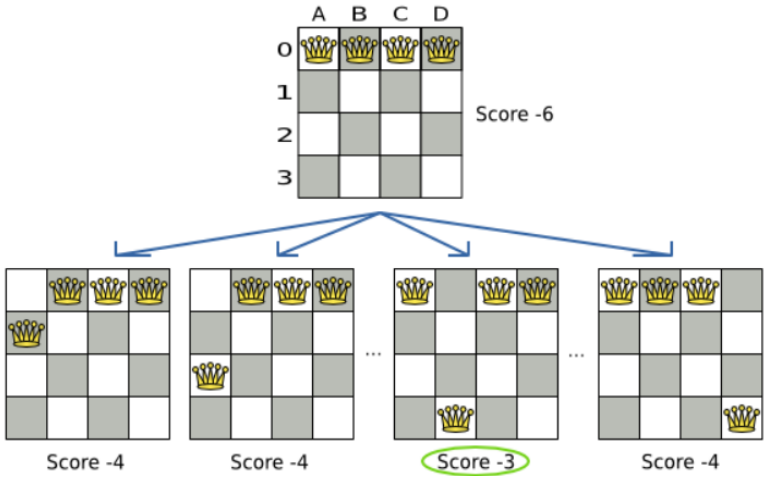

Рисунок 2.1 – Вирішення наступного кроку на кроці 0 (приклад із 4 ферзями)

Оскільки хід B0 до B3 має найвищий бал (-3), він вибирається як наступний крок. Якщо кілька ходів мають однакову найвищу оцінку, один

вибирається випадковим чином, у цьому випадку від B0 до B3. Зауважимо, що від C0 до C3 також можна було вибрати, оскільки він також має оцінку -3.

Крок застосовується на розв'язку. З цього нового рішення локальний пошуковий розв'язувач пробує кожен хід знову, щоб вирішити наступний крок. Програма локального пошуку використовує не дерево пошуку, а шлях пошуку [16-17].

2.2.2 Алгоритм сходження на пагорб (англ. hill climbing)

Протягом багатьох років генетичні алгоритми та методи сходження на пагорб використовувалися як методи евристичної оптимізації для некриптографічних програм. Наприклад, форма сходження на пагорб була використана ще в 1961 році Мінським для розробки систем штучного інтелекту. У 1973 році Голланд представив концепцію генетичних алгоритмів, засновану на застосуванні версії дарвінівського принципу «виживання найпристосованішого» для дослідження клітинного автомату.

З кінця 1990-х років було доведено, що методи сходження на пагорб ефективні в дослідженнях криптографічного характеру. Значна кількість важливих досліджень була проведена щодо використання алгоритму сходження на пагорб для оптимізації криптографічних властивостей як булевих функцій так і S-блоків. Сходження на пагорб також використовувався в поєднанні з іншими евристичними методами, такими як моделювання відпалу і генетичні алгоритми. Основна техніка сходження на пагорб передбачає пошук на кожній ітерації елементів функції, які потрібно змінити, що призведе до покращення вже отриманих результатів. Наприкінці процесу очікується, що кінцевий результат представлятиме найкраще доступне рішення.

Процес сходження на пагорб вимагає визначення відповідної функції, щоб оптимізувати одну або кілька конкретних криптографічних властивостей. Функція придатності є мірою цільової властивості, і рішення прийняти або відхилити прогресивні вихідні функції базується на цій мірі. Значна частина

роботи, яка була виконана з оптимізації властивостей булевої функції за допомогою цієї евристичної техніки, використовувала функцію відповідності, визначену для оцінки міри нелінійності. Для методу сходження на пагорб, описаного вище, ключовим кроком у процесі є заміна двох окремих елементів у таблиці істинності функції, щоб ітераційні покращення могли бути зроблені до цільової властивості. Нелінійність є найбільш природною мірою придатності для цього процесу через прямий вплив такого обміну на перетворення Уолша-Адамара булевої функції та, отже, на відповідне значення нелінійності. Зокрема, зміна на один біт у таблиці істинності булевої функції призводить до зміни перетворення Уолша Адамара функції ∈ {±2}, змушуючи міру нелінійності змінюватися на {±1}. Подібним чином, зміна перетворення Уолша Адамара булевої функції ∈ {0,±4} в результаті двобітової зміни в таблиці істинності функції змушує змінювати міру нелінійності ∈ {0,±2} [18-19].

1) Визначається *fit(f)* як функція відповідності булевої функції *f*.

2) Створюється булева функція випадкової *N*-змінної, $f(x), x = 0, \dots, 2^N - 1$.

3) Ітерація:

– Набори вдосконалення форми *I(f)* і обчислення *fit(f)*.

– Обираються дві позиції біта, *i* та *j*, де $f(i) \neq f(j)$.

– Отримаємо функцію-кандидата $g(x) = f(x)$, де $x = 0, \dots, i - 1, i + 1, \dots, j - 1, j + 1, \dots, 2^N - 1$; і $g(i) = f(i) \oplus 1$, $g(j) = f(j) \oplus 1$.

– Набори вдосконалення форми *I(g)* і обчислення *fit(g)*. Якщо $\forall$ ø $\in I(g)$, $|WHT(ø)| < WHT_{max}$, тоді нехай $f = g$.

– Якщо цикл досягає ліміту ітерацій без подальших покращень, вихід.

– Вихідна кінцева булева функція g, що представляє найкращу досяжну придатність.

4) Кроки 2, 3 повторюються за потреби, щоб піднятися на пагорб із потрібною кількістю булевих функцій.

2.2.3 Генетичні алгоритми

Генетичні алгоритми вперше були використані для криптографічних цілей на початку 1990-х років як інструменти для криптоаналізу класичних шифрів. Зовсім недавно генетичні алгоритми застосовувалися з подібними цілями, як і метод сходження на пагорб, для генерації булевих функцій, намагаючись удосконалити ряд важливих криптографічних властивостей, які, як відомо, забезпечують більшу безпеку.

Концепція, що лежить в основі генетичних алгоритмів, включає механізм еволюції, відомий як природний відбір. У процесі природного відбору популяція батьків схрещується для народження дітей. Можуть відбутися певні мутації, а потім процес відбору, коли лише найпристосованіші особини виживають, щоб стати наступним поколінням. Генетичні алгоритми приймають цю принципову ідею, яка є фундаментальною для теорії еволюції.

Подібно до методу сходження на пагорб, генетичні алгоритми використовують функцію пристосованості, щоб визначити міру пристосованості індивідуумів у популяції. Лише найкращі рішення вибираються для продовження в наступному поколінні.

Нехай T – кількість початкових стартових функцій. Опис базового генетичного алгоритму для покращення криптографічних властивостей булевих функцій наведено в алгоритмі [20-21]:

1) Визначається *fit(f)* як функція відповідності булевої функції *f*.

2) Визначається функція розведення двох булевих функцій зі змінною *N*, *f* і *g*, як *breed(f, g)*.

3) Створюється набір T випадкових логічних функцій N-змінних, що представляють батьківський пул, P_i, де $i = 1, .., T$.

4) Ітерація:

- Генерується набір $\frac{T(T-1)}{2}$ булевих функцій (нащадків), $C_i = breed(P_j, P_k)$, де $j = 1, \dots, T$, $k = 1, \dots, T$, $j \neq k$.
- Формується відсортований комбінований набір $S = P \cup C$, де $S = \left\{S_1, \dots, S_{T+\frac{T(T-1)}{2}}\right\}$ і $fit(S_l) \geq fit(S_{l+1})$, $l = 1,2, \dots, T + \frac{T(T-1)}{2} - 1$.
- Зберігайте найкращі функції T, замінюючи новими $P_i = S_i$, де $i = 1, \dots, T$.
- Скидання: необов'язково.
- Ітерація, доки не буде виконано заданий критерій зупинки.

5) Кроки 3, 4 повторюються за потреби, щоб обчислити генетичний алгоритм для бажаної кількості початкових пулів.

2.2.4 Алгоритм імітації відпалу (англ. simulated annealing)

У 1983 році було запропоновано імітований відпал, новий метод пошуку, натхненний процесами охолодження розплавлених металів. Він поєднує сходження на пагорб із імовірнісним прийняттям недосконалих ходів.

Пошук починається з деякого початкового стану *S := S_0*. Існує контрольний параметр *T*, відомий як температура. Це починається «високо» з T_0 і поступово знижується. При кожній температурі робиться спроба *MIL* (переміщення у внутрішньому циклі) переходів до нових станів. Стан-кандидат *Y* вибирається випадковим чином з *N(S)* поточного стану. Розраховується зміна значення δ з *f*. Якщо це покращує значення *f(S)* (тобто якщо $\delta < 0$ для задачі мінімізації), тоді здійснюється перехід до цього стану (*S = Y*); якщо ні, то береться з певною ймовірністю. Чим гірший крок, тим менша ймовірність, що його приймуть. Чим нижча температура T, тим менша ймовірність того, що буде прийнято погіршення. Імовірнісна прийнятність визначається генеруванням випадкового значення *U* в діапазоні (0...1) і виконанням зазначеного порівняння. Спочатку температура висока, і прийнятні практично будь-які ходи.

Коли температура опускається, стає все важче приймати погіршення. Зрештою, дозволяються лише покращення ходів, і процес стає "замороженим". Алгоритм закінчується, коли виконується критерій зупинки. Загальні критерії зупинки і ті, що використовуються для роботи в цій роботі, повинні припинити пошук після того, як було виконано фіксоване число *MaxIL* внутрішніх циклів. Або ж тоді, коли було виконано деяке максимальне число *MUL* послідовних непродуктивних внутрішніх циклів (тобто, без єдиного ходу, який був прийнятий). Загалом же записується найкращий стан, досягнутий поки що (оскільки пошук може насправді вийти з нього і згодом бути не в змозі знайти стан подібної якості). В кінці кожної внутрішнього циклу температура опускається. Найпростіший спосіб пониження температури — помножити на постійний коефіцієнт охолодження α в діапазоні (0…1); це відоме як геометричне охолодження [22-23].

$S := S_0$

$T := T_0$

repeat

{

for (int i = 0; i < *MIL*; i + +)

{

select $Y \in N(S)$

$\delta := f(Y) - f(S)$

if (δ < 0) then

$S := Y$

else

generate U := *rnd*(0, 1) if (U < exp($-\delta/T$)) then $S := Y$

}

$T = T \times \alpha$

}

until

Цикл повторюється поки не буде виконано критерій зупинки.

2.2.5 Метод градієнтного спуску (англ. Gradient Descent Method)

В загальному випадку можна розглядати пошук стаціонарної точки як таку, що має дві складові: напрямок і розмір кроку. Напрямок вирішує, який напрямок ми шукаємо далі, і розмір кроку визначає, як далеко ми йдемо в цьому конкретному напрямку. Такі методи можуть бути загалом описані як стартові в деякій довільній точці $x^{(0)}$ і потім на кожному кроці $k \geq 0$ ітеративно рухаючись при напрямку $\Delta x^{(k)}$ за величиною кроку t_k до наступної точки $x^{(k+1)} = x^{(k)} + t_k \cdot \Delta x^{(k)}$. При градієнтному спуску напрямок, який шукається, є від'ємним градієнтом в точці, тобто $\Delta x = -\nabla f(x)$. Таким чином, ітеративний пошук градієнтного спуску можна описати через наступне рекурсивне правило:

$$x^{(k+1)} = x^{(k)} - t_k \cdot \nabla \mathrm{f}(x^{(k)})$$

Вибір розміру кроку. Враховуючи, що пошук стаціонарної точки в даний час знаходиться в певній точці $x^{(k)}$, як нам вибрати наш розмір кроку t_k? Оскільки наша мета полягає в тому, щоб мінімізувати функцію, один підхід полягає в тому, щоб вибрати розмір кроку таким чином, що буде мінімізувати значення нової точки, тобто знайти розмір кроку, який мінімізує $f(x^{(k+1)})$. Оскільки $x^{(k+1)} = x^{(k)} - t\nabla \mathrm{f}(x^{(k)})$ розмір кроку t_k^* такого підходу: $t_k^* = \mathrm{argmin}_{t \geq 0} f(x^{(k)} - t\nabla \mathrm{f}(x^{(k)}))$ для тепер будемо вважати, що t_k^* може бути обчислено аналітично, а пізніше повторно відвідати це припущення.

Формально, враховуючи бажану точність > 0, визначається градієнтний спуск [24]:

Нехай $x^{(0)}$, $k \leftarrow 0$

while $\|\nabla f(x^{(k)})\| \geq \epsilon$ do

$x^{(k+1)} = x^{(k)} - t_k \nabla f(x^{(k)})$

$k \leftarrow k + 1$

end while

return $x^{(k)}$

3 АНАЛІЗ ПРАКТИКИ ЗАСТОСУВАННЯ ЕВРИСТИЧНИХ АЛГОРИТМІВ ПОШУКУ ВИСОКОНЕЛІНІЙНИХ S-БЛОКІВ

3.1 Критерії і показники, необхідні для практики реалізації алгоритмів пошуку високонелінійних S-блоків

Симетричні криптографічні примітиви широко використовуються завдяки їх високій продуктивності та низькій складності реалізації. При гарантуванні конфіденційності за допомогою блочних шифрів симетричні примітиви використовуються для забезпечення цілісності інформації на основі кодів автентифікації повідомлень і геш-функцій, як компоненти електронного цифрового підпису для захисту автентичності, генерації псевдовипадкових послідовностей, як частина протоколів автентифікації тощо. Відповідно до відомих принципів Шеннона, такі алгоритми використовують нелінійні операції для плутанини та лінійні перетворення для дифузії. Послідовне багаторазове застосування плутанини та дифузії забезпечує високий рівень криптографічної міцності. Вузли нелінійної підстановки для сучасних симетричних примітивів зазвичай реалізуються як таблиці підстановки або S-блоки.

Враховуючи, що більшість сучасних блокових алгоритмів використовують одну лінійну операцію додавання за модулем 2 для введення раундових ключів і комбінування міжраундових значень, S-блок є єдиним елементом, який визначає нелінійність трансформацій шифрування та рівень його стійкості до криптоаналітичних атак.

Необхідну кількість раундів блокових шифрів розраховано виходячи із забезпечення стійкості до відомих видів криптографічного аналізу за умови заданих властивостей вузлів нелінійної підстановки. Багато потокових алгоритмів, криптографічних геш-функцій і генераторів псевдовипадкової

послідовності засновані на блокових шифрах або їх структурних елементах. Таким чином, криптографічна міцність більшості сучасних симетричних примітивів значною мірою залежить від властивостей використовуваних S-блоків.

Найбільш просунуті ключові криптосистеми, засновані на ідеї виробництва шифру, яка вже стала традиційною і представляють собою клас криптосистем, які повторюють складну операцію перетворення відкритого тексту в зашифрований текст. Кожне таке повторення (ітерація) відоме як цикл шифру. Складна (складена) операція, що виконується в кожному циклі, зазвичай є комбінацією набору примітивних операцій, таких як зсув, лінійне перетворення, додавання за модулем і заміна. У відповідному поєднанні ці перетворення повинні реалізовувати концепцію побудови таких шифрів, яка, як відомо, полягає в тому, що комбінація операцій перестановки та заміни окремо слабких перетворень може призвести до криптографічно сильного нелінійного перетворення, якщо застосувати достатню кількість разів. Операції підстановки в багатьох шифрах виступають у цьому випадку основним нелінійним елементом циклічного перетворення.

Нелінійність елементів, що складають симетричні блочні криптосхеми, має великий вплив на міцність підсистеми захисту інформації в сучасних телекомунікаційних системах. Розглядається вибір якісних замінників (S-блоків). На сьогоднішній день в існуючих інструментах оцінки міцності підстановок немає жодного методу визначення найкращого S-блоку з точки зору протидії різним криптоаналітичним атакам і технікам його апаратної та / або програмної реалізації. У сучасних алгоритмах замість фіксованих підстановок використовуються лінійні перетворення для відображення встановленого S-блоку (часто єдиного) в інший блок із достатньо великого набору еквівалентів.

Саме тому значні зусилля дослідників спрямовані на вивчення властивостей і конструювання підстановок з високими криптографічними показниками. Одним із найпопулярніших для опису та дослідження

властивостей S-блоків є математичний апарат лінійної алгебри і, зокрема, апарат булевих функцій.

S-блок лежить в основі будь-якого блокового шифру та є джерелом нелінійності. Вивчення шифру на міцність, як правило, починається з вивчення властивостей його S-блоків. Загалом, S-блок m×n відображає $Zm2 \rightarrow Zn2$.

Найпростішим способом визначення S-блоків є таблиця значень, яка вказує, куди йде кожна бітова комбінація. Розміри пам'яті, необхідні для зберігання такої таблиці, такі:

- $4 \times 4 = 16$
- $8 \times 8 = 256$
- $16 \times 16 = 65536$
- $32 \times 32 = 4294967296 \sim 4 \times 109$
- $64 \times 64 = 18446744073709551616 \sim 2 \times 1019$

Таблиці є зручними для програмної реалізації, але накладаються обмеження на їх розмір. Існує багато конкуруючих підходів до вибору S-блоків, серед яких можна виділити чотири основні:

- Випадкова вибірка. Зрозуміло, що малі випадкові S-блоки не є надійними, але великі випадкові S-блоки можуть бути достатньо хорошими. Випадкові S-блоки з вісьмома або більше входами можуть бути досить сильними. Потужність S-блоків підвищується, коли вони є випадковими та залежними від ключа.
- Відбір проб з подальшим тестуванням. У деяких шифрах спочатку генеруються випадкові S-блоки, а потім перевіряються їх властивості на відповідність вимогам.
- Ручна розробка. У ній рідко використовується математичний апарат: S-блоки створюються за допомогою інтуїтивно зрозумілих прийомів.
- Математичний розвиток. S-блоки створюються за законами математики, саме тому вони мають гарантовану стійкість до диференціального

та лінійного криптоаналізу та хороші дифузійні властивості. Були пропозиції поєднати «математичний» і «ручний» підходи, але на практиці конкурують випадково вибрані S-блоки та S-блоки з певними властивостями. Переваги останнього підходу включають оптимізацію проти відомих методів атак – диференціального та лінійного криптоаналізу.

В останні роки з'явилося багато підходів до отримання таблиць підстановки, наприклад, дробове лінійне перетворення, кубічне дробове перетворення, евристичний підхід, модульний підхід та інші. Показано спосіб отримання динамічного S-блоку шляхом застосування перетворень підстановки-перестановки до вхідного значення S-блоку. Цей (розглянутий) S-блок структурно відрізняється від розробленого S-блоку; запропонований алгоритм являє собою метод отримання фіксованого S-блока.

Іншими словами, S-блок – відображення m-бітних входів на n-бітні виходи. S-блоки є частиною функції перетворення та важливі для надійності алгоритму шифрування. Будь-які зміни на вході S-блоку повинні призвести до аналогічних випадкових змін на виході. Залежність вихідних значень від вхідних не повинна бути лінійною або легко апроксимованою лінійними функціями (ця властивість використовується при застосуванні лінійного криптоаналізу).

3.2 Аналіз прикладів використання найефективніших алгоритмів пошуку високонелінійних S-блоків

Нагадаємо, що важливими характеристиками якості криптографічних перетворювань являються поняття критерію строгого лавинного ефекту (SAC – strict avalanche criterion) і критерію розповсюдження (PC – propagation criterion). Основна сутність їх полягає в оцінюванні імовірності зміни значення БФ в залежності від зміни частини бітів аргументів цих функцій.

Булева функція $f(x)$, $a \in GF(2)^n$ задовольняє:

– Критерій розповсюдження ступеня $1 \leq n\ (PC(l))$, якщо при заміні $1 \leq i \leq l$ довільних 75 вхідних бітів на свої доповнення функція f(x) змінюється з 0.5, що еквівалентно збалансованості різниці Δf (x,a) для будь-яких a, вага яких $1 \leq wt(a) \leq l$;

– Критерій розповсюдження ступеня $l \leq n$ і порядку $k \leq n - 1\ (PC(l)/k)$, якщо будь-яка функція, що отримана із БФ f(x) фіксацією будь-яких її k змінних, що задовольняють критерію розповсюдження.

Незважаючи на безліч існуючих рішень у сфері симетричної криптографії, актуальним є пошук підходів, що забезпечують захист від існуючих та запропонованих видів атак на алгоритми шифрування. Необхідно обґрунтувати критерії розробки та вдосконалення методів створення вузлів нелінійних перетворень, а також провести дослідження методів криптоаналізу та теорії векторних булевих функцій [25-26].

3.2.1 Дослідження та пошук оптимальних параметрів алгоритму сходження на пагорб

У числовому аналізі підйом на пагорб є технікою математичної оптимізації, яка належить до сімейства локального пошуку. Це ітераційний алгоритм, який починається з довільного вирішення проблеми, а потім намагається знайти краще рішення, вносячи поступові зміни в рішення. Якщо зміна призводить до кращого рішення, до нового рішення вноситься ще одна поступова зміна і так далі, доки не буде знайдено подальших покращень.

Наприклад, підйом на пагорб можна застосувати до задачі комівояжера. Легко знайти початкове рішення, яке відвідує всі міста, але, швидше за все, буде дуже поганим порівняно з оптимальним рішенням. Алгоритм починається з такого рішення та вносить у нього невеликі покращення, наприклад змінює порядок відвідування двох міст. Зрештою, ймовірно, буде отримано набагато коротший маршрут.

Підйом на пагорб знаходить оптимальні розв’язки для опуклих задач – для інших задач він знаходить лише локальні оптимуми (рішення, які не можуть бути покращені будь-якими сусідніми конфігураціями), які не обов’язково є найкращим можливим рішенням (глобальним оптимумом) з усіх можливих рішень (простір пошуку). Приклади алгоритмів, які розв’язують опуклі проблеми за допомогою підйому на пагорб, включають симплексний алгоритм для лінійного програмування та двійковий пошук. складні схеми, засновані на ітераціях (наприклад, ітерований локальний пошук), або на пам’яті (наприклад, реактивна оптимізація пошуку та пошук табу), або на стохастичних модифікаціях без пам’яті (наприклад, імітований відпал).

Першими (1-2) були досліджені параметри з роботи Кларка. Функція WHS була запропонована Кларком і має вигляд:

$$WHS = \sum_{b=1}^{255} \sum_{i=0}^{255} \left| ||WHT[b,i]|| - X \right|^{R} \tag{3.1}$$

де:

– WHT – Спектральні коефіцієнти Уолша-Адамара;

– i – цикл за всіма компонентними функціями та їх лінійними комбінаціями;

– b – цикл за всіма лінійними функціями;

– X and R – параметри з реальними значеннями.

Було здійснено кілька десятків запусків алгоритму сходження на пагорб з різними параметрами попарно, які були запропоновані Кларком в його роботі. У той же час, не було ніякої нелінійності 104. Щоб досягнути нелінійності N_f = 104 потрібна або набагато більша кількість ітерацій, або інший алгоритм пошуку, який виходить за рамки цього дослідження. У той же час, нелінійність 102 була досягнута досить часто і, як правило, після досягнення нелінійності 102, і наступних декількох ітерацій зменшення цільової функції, нелінійність знову загострилася до 100-96.

Основні та рекомендовані параметри які були задіяні в коді:

/*необхідна нелінійність*/

target_nonlinearity=104

/* цільова дельта рівномірність */

target_delta_uniformity=8

/*необхідний мінімальний ступінь*/

target_min_degree=7

/*необхідний алгебраїчний імунітет*/

target_algebraic_immunity=3

/*максимальна кількість спроб*/

max_try_count = 900000

/*кількість випадкових обмінів*/

changed_positions_count=1

/* найменування цінової функції*/

cost_function_name=power_max_wht

/* Параметри цінової функції*/

power_max_wht.x_param=36

power_max_wht.r_param=4

1) X=0; R=3

Для даних параметрів алгоритм виконав > 160 спроб, в кожній було > 10000 ітерацій. Максимальна нелінійність при даних параметрах, яка була досягнута, дорівнювала 102.

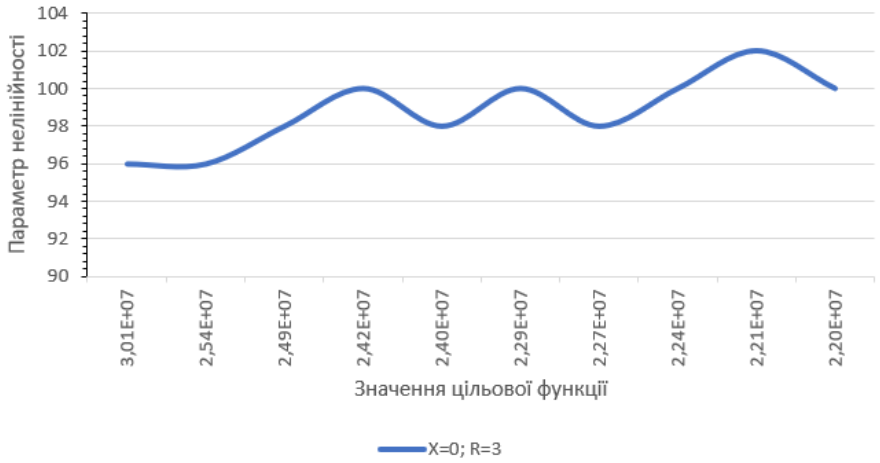

Рисунок 3.1 – Результат запуску з параметрами X=0; R=3

2) X=4; R=3

Для даних параметрів алгоритм виконав > 160 спроб, в кожній було > 10000 ітерацій. Максимальна нелінійність при даних параметрах, яка була досягнута, дорівнювала 102.

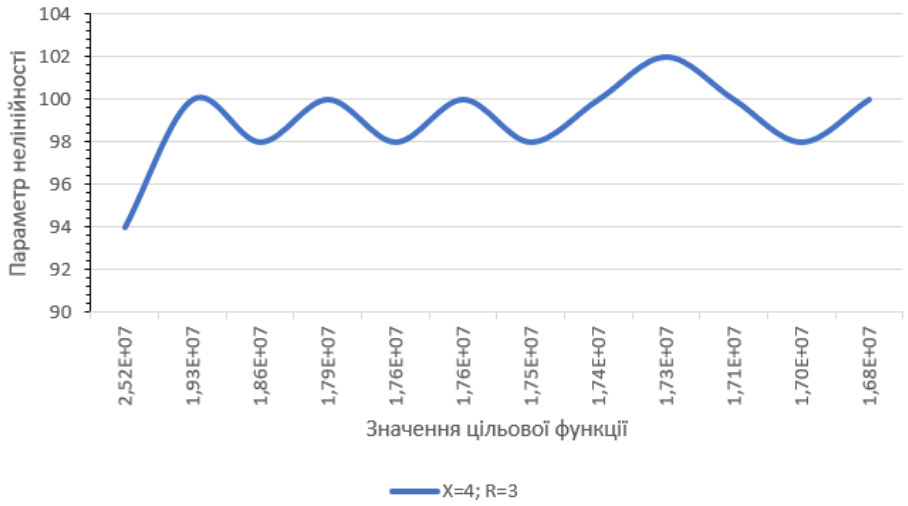

Рисунок 3.2 – Результат запуску з параметрами X=4; R=3

3) X=36; R=4

Для даних параметрів алгоритм виконав 19 спроб, в кожній було приблизно 8300 ітерацій. Розрахунок було завершено за 4 хвилини та досягнуто максимальної нелінійності в 104.

Рисунок 3.3 – Результат запуску з параметрами X=36; R=4

При X = 36; R = 4 параметрах в роботі [27] загалом було знайдено 16 980 (21,5 % від загальної кількості циклів тестувань) цільових S-блоків із середнім

часом пошуку одного блоку у 33,2 секунди. Гістограма розподілу кількості знайдених цільових S-блоків в залежності від часу, що був затрачений на їх пошук, наведено на Рисунку 3.4. На рисунку наведено значення, що не перевищують двох хвилин пошуку. Ще 390 цільових S-блоків (2,3 % від загальної кількості) було знайдено за час, що перевищував дві хвилини.

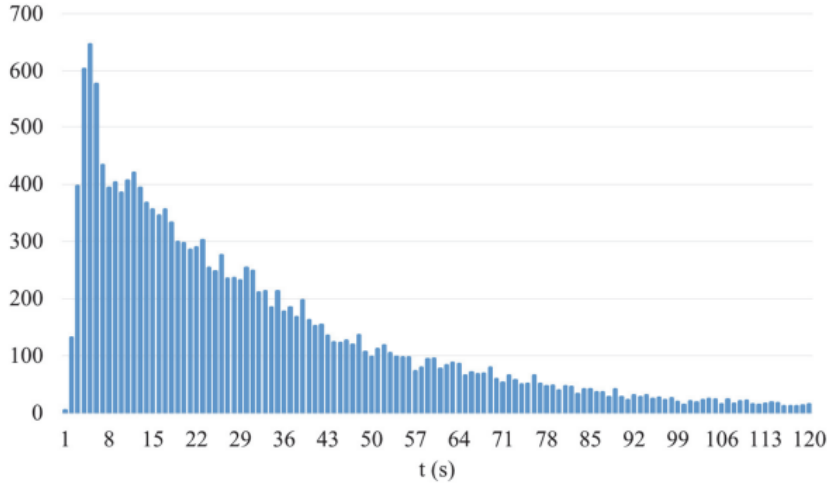

Рисунок 3.4 – Розподіл кількості знайдених цільових S-блоку в залежності від часу (t, секунди), що було затрачено на їх пошук

Враховуючи кратність коефіцієнтів спектру Уолша-Адамара чотирьом та перевагу застосування 64 бітних чисел, у [28] досліджено дещо модифіковану функцію виду:

$$WHS = \sum_{b=1}^{255}\sum_{i=0}^{255}\left|\frac{|WHT[b,i]|-X}{4}\right|^{R} \qquad (3.2)$$

З метою пошуку параметрів X та R було проведено їх зміну у більш широкому діапазоні ніж це наведено у роботі Кларка. При кожному окремому іспиті (окремому запуску алгоритму сходження на пагорб) фіксувалась кількість ітерацій алгоритму пошуку до знаходження бієктивного S-блоку з нелінійністю 104 або факт невдалого пошуку (виконання інших критеріїв зупинки пошуку).

Зауважмо, що ймовірно іншими алгоритмами можна досягти бажаного результату за меншу кількість ітерацій. Однак інший алгоритм потребує унікальних налаштувань під обрану цільову функцію та може бути не

оптимальним, у зв’язку з чим, буде важко проводити порівняння ефективності (кількості ітерацій) різних цільових функцій.

3.2.2 Дослідження та пошук оптимальних параметрів алгоритму імітації відпалу

Даний алгоритм проаналізовано на основі параметру α (0,6; 0,7; 0,8; 0,9; 0,95), який є параметром зміни температури (зменшення температури).

Загальна температура варіюється від значення, де ймовірність вирішення гіршого рішення була майже 0 до більшого одного, куди ймовірність була наближена до 1. Збільшення в T_0 було зроблено відповідно до правила:

$$T_0^{i+1} = 1{,}13 \cdot T_0^i \tag{3.3}$$

Основні та рекомендовані параметри які були задіяні в коді:

```
/*необхідна нелінійність*/
target_nonlinearity=104
/*необхідна дельта рівномірність*/
target_delta_uniformity=12
/*необхідний мінімальний ступінь*/
target_min_degree=6
/*необхідний алгебраїчний імунітет*/
target_algebraic_immunity=3
/*обмеження зверху на значення цільової функції*/
limit=3000000
/*максимальна кількість зовнішніх циклів*/
max_outer_loops=50
/*максимальна кількість внутрішніх циклів (у кожному потоці)*/
max_inner_loops=3000
/*максимальна кількість бездіяльних циклів*/
max_frozen_outer_loops=2
```

/*початкова температура*/

initial_temperature=10.0

/*швидкість зміни температури*/

alpha_parameter=0.6

/*ідентифікатор цінової функції*/

cost_function_name=power_max_wht

/* параметри цінової функції*/

power_max_wht.x_param=36

power_max_wht.r_param=4

1) $\alpha = 0,6$

В результаті першого запуску було виконано 13 спроб, що зайняло 6 хвилин. В результаті досягнуто максимальної нелінійності в 102, температура, яка частіше фігурувала та визначена на Рисунку 3.5: T=10, T=6, T=3,6, T=2, 16.

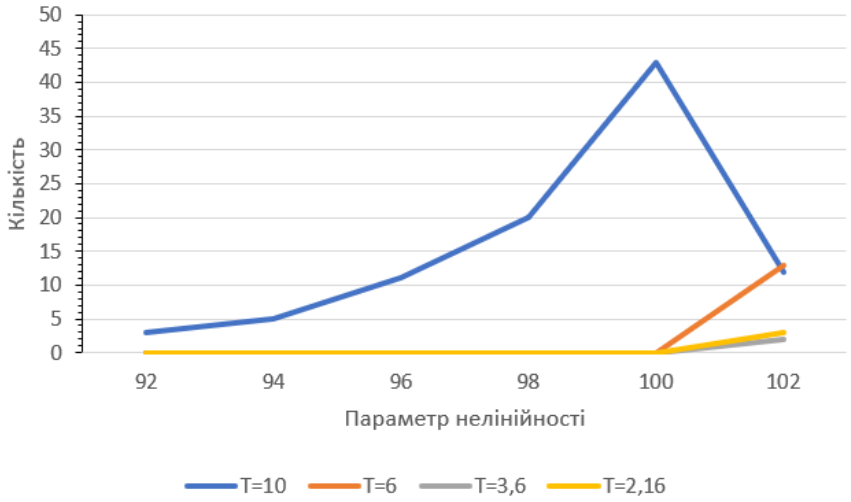

Рисунок 3.5 – Результат запуску для $\alpha = 0,6$

2) $\alpha = 0,7$

В результаті другого запуску було виконано 17 спроб, що зайняло 14 хвилин. В результаті досягнуто максимальної нелінійності в 102, температура, яка частіше фігурувала та визначена на Рисунку 3.6: T=10, T=7, T=4,9, T=2,4, T=1,68, T=1,18, T=0,58.

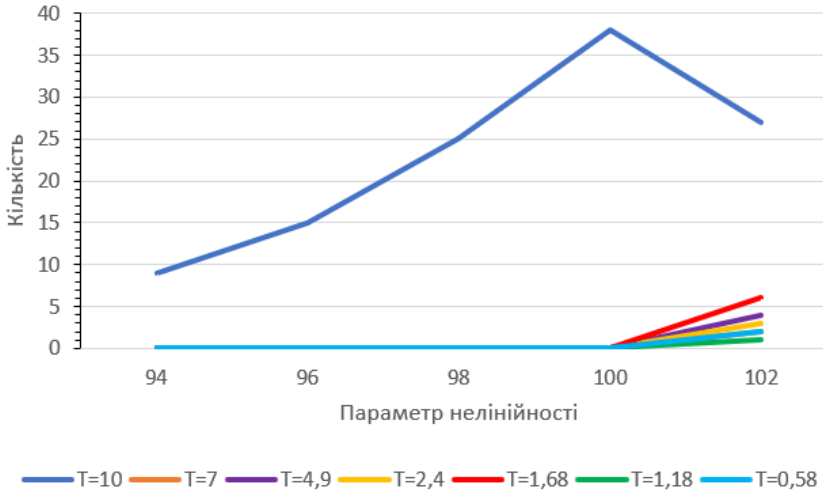

Рисунок 3.6 – Результат запуску для α = 0,7

3) α = 0,8

В результаті другого запуску було виконано 1 спроба, що зайняло 1 хвилину. В результаті досягнуто максимальної нелінійності в 102, температура, яка частіше фігурувала та визначена на Рисунку 3.7: T=10, T=8, T=6,4, T=5,12, T=3,28.

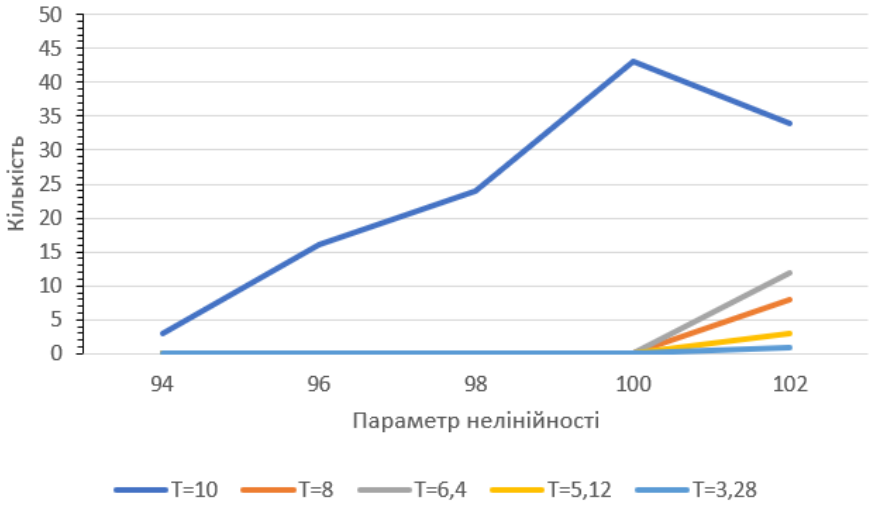

Рисунок 3.7 – Результат запуску для α = 0,8

4) α = 0,9

В результаті другого запуску було виконано 1 спроба, що зайняло 0,5 хвилин. В результаті досягнуто максимальної нелінійності в 102, температура, яка частіше фігурувала та визначена на Рисунку 3.8: T=10, T=9, T=8,1, T=7,29, T=6,56.

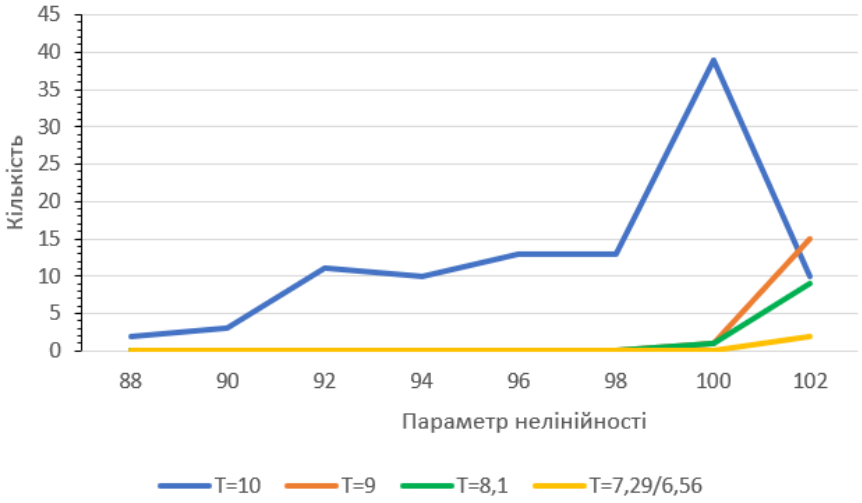

Рисунок 3.8 – Результат запуску для α = 0,9

5) α = 0,95

В результаті другого запуску було виконано 1 спроба, що зайняло 0,7 хвилин. В результаті досягнуто максимальної нелінійності в 102, температура, яка частіше фігурувала та визначена на Рисунку 3.9: T=10.

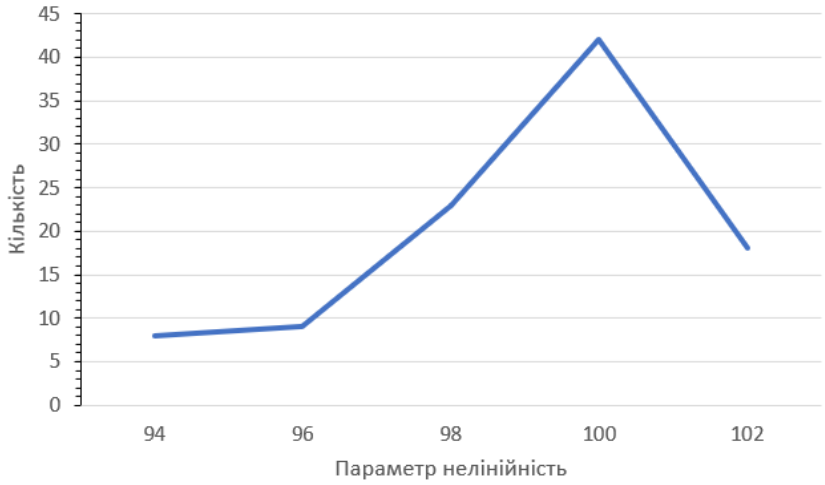

Рисунок 3.9 – Результат запуску для α = 0,95

Загальний графік результату досліджень зображено на Рисунку 3.10, з якого видно, що максимальна нелінійність, яка була досягнута = 102 для всіх α з проміжку {0,6;0,95}.

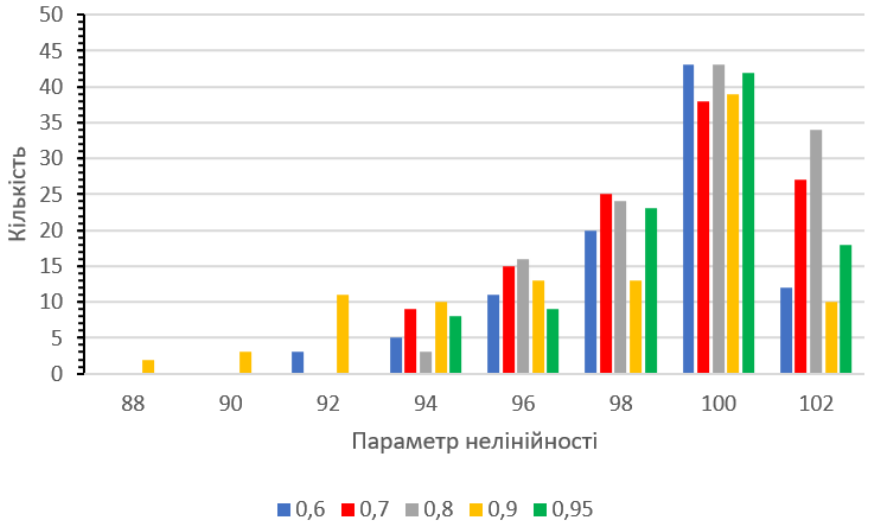

Рисунок 3.10 – Результат запуску для α = 0,95

Виходячи з результатів досліджень, робимо висновок, що метод імітації відпалу робить хорошу роботу з пошуку цілі (тобто з заданими властивостями) S-блоку. 100% ймовірність знаходження S-блоку не є оптимальним шляхом з точки зору часу, витраченого на пошук. Введення додаткових обмежень зменшує час, витрачений на кожну спробу, але також зменшує ймовірність знаходження цільової S-скриньки в кожній спробі. Таким чином, результати пошуку за допомогою методу симульованого відпалу дуже чутливі до всіх параметрів вхідного пошуку, а їх оптимізація є дуже трудомістким процесом. Досліджено вплив вхідних параметрів симульованого методу відпалу на результат пошуку цільового S-блок. На основі результатів дослідження представлені порівняльні характеристики часу пошуку і внутрішніх станів алгоритму [29].

3.2.3 Дослідження та пошук оптимальних параметрів генетичних алгоритмів

Опис функції вартості:

Для проведення досліджень впливу параметрів на значення функції вартості здійснено тестування. При цьому параметр end змінювали у діапазоні від 0 до 32 з кроком 4.

Функція вартості WCF у загальному випадку має наступний вигляд:

$$WCF = \sum_{b=1}^{255} \sum_{i=0}^{255} \prod_{\substack{j=start \\ j+=step}}^{end} \left| |WHT[b,i]| - j \right| \quad (3.4)$$

де

– WHT–спектральні коефіцієнти Уолша–Адамара;

– start, step, end – деякі цілі значення, як правило start = 0, step = 4 (виходячи з кратності коефіцієнтів WHT чотирма);

– i – змінна циклу за всіма компонентними функціями та їх лінійними комбінаціями;

– b – змінна циклу за всіма лінійними функціями.

Функція WCF фактично приймає за нульовий вклад значень спектральних коефіцієнтів Уолша–Адамара та враховує лише їх крайні значення, індекси яких за модулем більші ніж значення end [30].

Основні та рекомендовані параметри які були задіяні в коді:

/*необхідна нелінійність*/

target_nonlinearity=104

/*необхідна дельта рівномірність*/

target_delta_uniformity=10

/*необхідний мінімальний ступінь*/

target_min_degree=6

/*необхідний алгебраїчний імунітет*/

target_algebraic_immunity=3

/*обмеження зверху на значення цільової функції*/

limit=3943006470144000

/*кількість мутацій кожного нащадка*/

mutations_per_parent=10

/*кількість нащадків кожного предка*/

child_per_parent=10

/*максимальна кількість поколінь*/

iterations_count=15000

/* компаратор */

comparator = sbox_compare104_random

/*метод вибору найкращих представників*/

selection_method = basic_selection_method

/*метод кросовера*/

crossover_method = None

crossover.child_count = 1

crossover.probability = 0.1

/* найменування цінової функції*/

cost_function_name=wcf

/* параметри цінової функції*/

power_max_wht.x_param=40

power_max_wht.r_param=3

wcf.start=0

wcf.step=4

wcf.end = 32

1) end=16

Під час запуску було виконано 6400+ ітерацій, це зайняло 8 хвилин. Максимальна нелінійність, яка була досягнута при даному значенні параметру становить 102 (Рисунок 3.11).

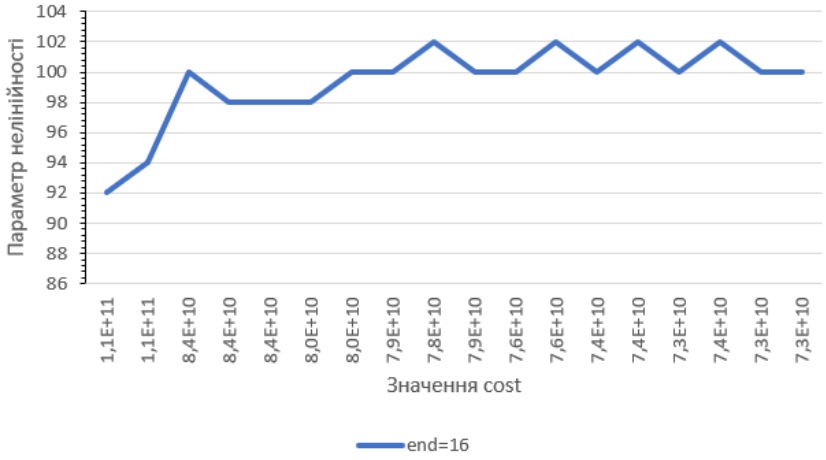

Рисунок 3.11 – Результат дослідження при end=16

2) end=20

Під час запуску було виконано 1000+ ітерацій, це зайняло 2,7 хвилини. Максимальна нелінійність, яка була досягнута при даному значенні параметру становить 102 (Рисунок 3.12).

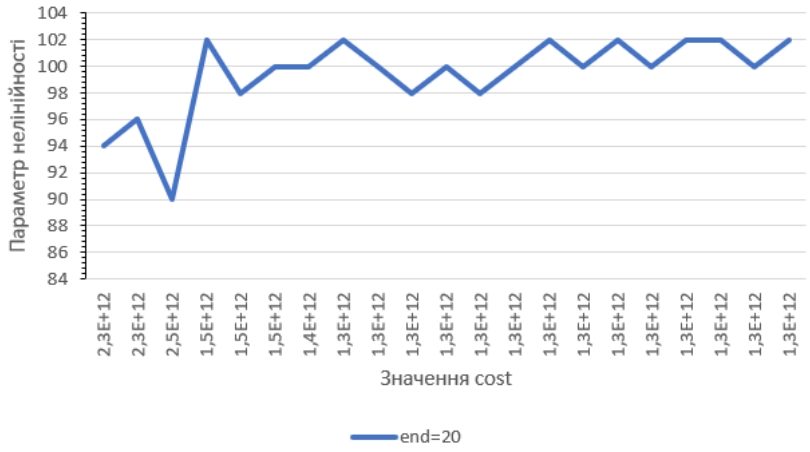

Рисунок 3.12 – Результат дослідження при end=20

3) end=24

Під час запуску було виконано 1300+ ітерацій, це зайняло 2,2 хвилини. Максимальна нелінійність, яка була досягнута при даному значенні параметру становить 102 (Рисунок 3.13).

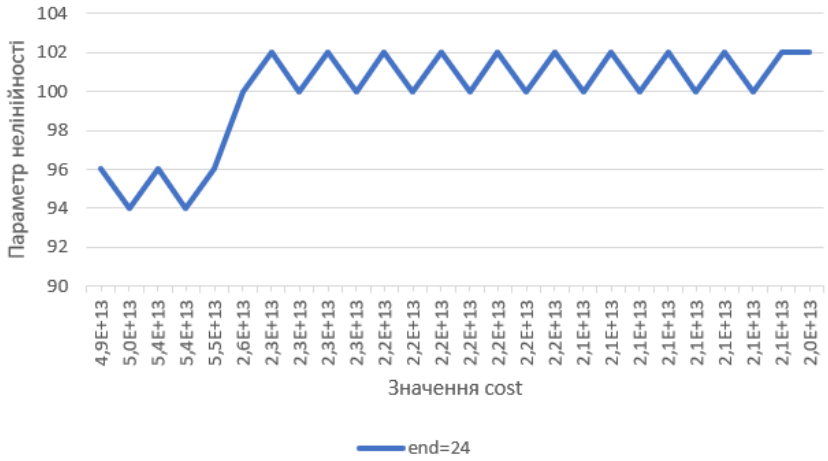

Рисунок 3.13 – Результат дослідження при end=24

4) end=28

Під час запуску було виконано 1000+ ітерацій, це зайняло 1,28 хвилини. Максимальна нелінійність, яка була досягнута при даному значенні параметру становить 102 (Рисунок 3.14).

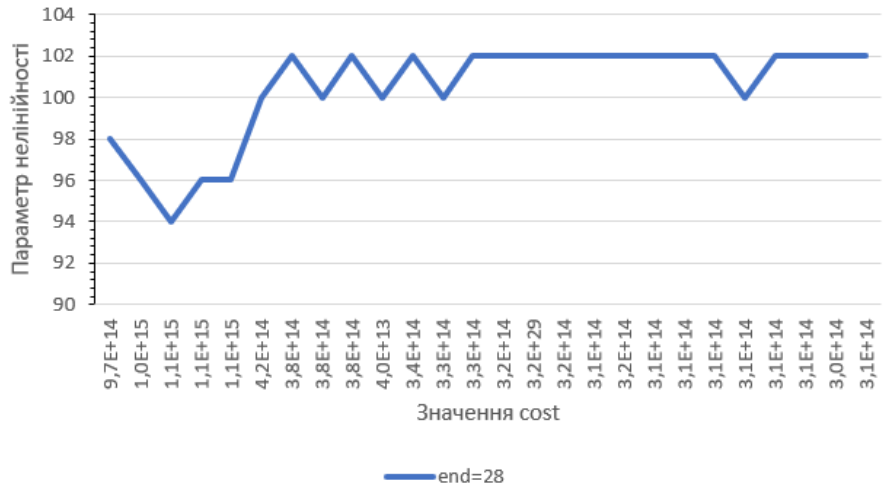

Рисунок 3.14 – Результат дослідження при end=28

5) end=32

Під час запуску було виконано 6000+ ітерацій, це зайняло 3,5 хвилини. Максимальна нелінійність, яка була досягнута при даному значенні параметру становить 104 (Рисунок 3.15).

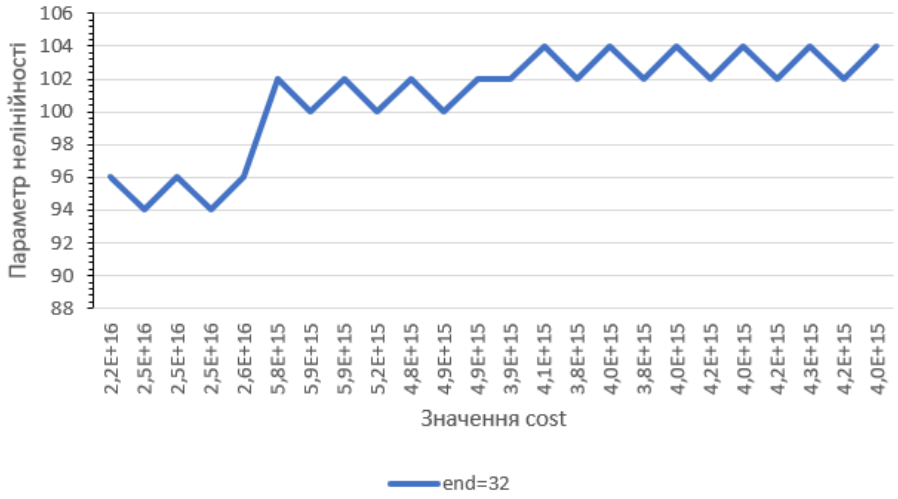

Рисунок 3.15 – Результат дослідження при end=32

Отже, при значенні параметру end=32, була досягнута бажана нелінійність 104. Крім того, функція вартості WCF дійсно дозволяє значно прискорити формування високонелінійних підстановок.

Вже при end = 40 значення функції WCF перевищує 64-бітне значення та потребує застосування більш довгої арифметики, що потенційно зменшує продуктивність роботи алгоритму пошуку [30].

3.2.4 Дослідження та пошук оптимальних параметрів методу градієнтного спуску

У математиці, градієнтний спуск (також називається найкрутішим спуском) – це ітеративний алгоритм для оптимізації першого порядку знаходження локальних мінімумів диференційованих функцій. Ідея полягає в тому, щоб виконати ітерацію в протилежному напрямку до градієнтної функції (або приблизної зміни градієнта) в поточній точці, тому що це напрямок найкрутішого спуску. Натомість, кроки в напрямку градієнта призводять до локального максимуму цієї функції. Ця процедура називається градієнтним сходженням.

У статті [34] оптимальна заміна відноситься до перестановки з:

– Максимальний алгебраїчний степінь;

– Максимальна алгебраїчна імунітет при мінімальній кількості рівнянь;

– Максимальні значення δ-рівномірності та нелінійності;

– Відсутність нерухомих точок.

Перший варіант – класичний алгоритм градієнтного спуску, завдання полягало в тому, щоб знайти кілька нееквівалентних підстановок з нелінійністю, що дорівнює або перевищує 100. Програма згенерувала випадкову перестановку та перевірила її на оптимальність. Після 12 годин роботи кластера було знайдено 27 оптимальних перестановок, чотири з яких були нееквівалентними. Ця заміна має наступні характеристики:

– Нелінійність 100;

– Абсолютне значення автокореляції 96;

– Мінімальний алгебраїчний ступінь 7;

– 8-uniform;

– Алгебраїчна імунітет: система з 441 рівняння 3-го ступеня.

Крім того, в роботі проводився пошук підстановки з вищою нелінійністю. Однак для класичного алгоритму це виявилось неможливим. З

практичної точки зору генерація таких перестановок обчислювально надзвичайно складна.

Однак, якщо модифікувати функцію, то стає можливим досягнути вищу нелінійність. В роботі [31] запропоновано новий метод генерації підстановок на основі твердження:

Нехай $F: F_{2^n} \to F_{2^n}$. G:

$$\begin{cases} G(p_1) = F(p_2) & p_1 \neq p_2 \\ G(p_2) = F(p_1) & \\ G(p_x) = F(x) & x \notin \{p_1, p_2\} \end{cases} \tag{3.5}$$

Тоді:

$$\begin{gathered} \delta(F) - 4 \leq \delta(G) \leq \delta(F) + 4 \\ NL(F) - 2 \leq NL(G) \leq NL(F) + 2 \end{gathered} \tag{3.6}$$

Функція нелінійності (Nl) довільної векторної функції F обчислюється наступним чином:

$$NL(F) = 2^{n-1} - \frac{1}{2} \cdot max_{u \neq 0, v \in F_{2^n}} |\lambda(u, v)| \tag{3.7}$$

Алгоритм приймає як вхідні дані бієктивну векторну булеву функцію F з мінімальним значенням δ-рівномірності та кількістю значень (NP) у функції, які мають змінювати під час оптимізації криптографічних параметрів. Експериментально знайдено значення NP, що дорівнює 26, для n = 8 із забезпеченням необхідних властивостей заміщення.

- Нелінійність 104;
- Абсолютне значення автокореляції 80;
- Мінімальний алгебраїчний ступінь 7;
- 8-uniform;
- Алгебраїчна імунітет: система з 441 рівняння 3-го ступеня.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	68	8D	CA	4D	73	4B	4E	2A	D4	52	26	B3	54	1E	19	1F
1	22	03	46	3D	2D	4A	53	83	13	8A	B7	D5	25	79	F5	BD
2	58	2F	0D	02	ED	51	9E	11	F2	3E	55	5E	D1	16	3C	66
3	70	5D	F3	45	40	CC	E8	94	56	08	CE	1A	3A	D2	E1	DF
4	B5	38	6E	0E	E5	F4	F9	86	E9	4F	D6	85	23	CF	32	99
5	31	14	AE	EE	C8	48	D3	30	A1	92	41	B1	18	C4	2C	71
6	72	44	15	FD	37	BE	5F	AA	9B	88	D8	AB	89	9C	FA	60
7	EA	BC	62	0C	24	A6	A8	EC	67	20	DB	7C	28	DD	AC	5B
8	34	7E	10	F1	7B	8F	63	A0	05	9A	43	77	21	BF	27	09
9	C3	9F	B6	D7	29	C2	EB	C0	A4	8B	8C	1D	FB	FF	C1	B2
A	97	2E	F8	65	F6	75	07	04	49	33	E4	D9	B9	D0	42	C7
B	6C	90	00	8E	6F	50	01	C5	DA	47	3F	CD	69	A2	E2	7A
C	A7	C6	93	0F	0A	06	E6	2B	96	A3	1C	AF	6A	12	84	39
D	E7	B0	82	F7	FE	9D	87	5C	81	35	DE	B4	A5	FC	80	EF
E	CB	BB	6B	76	BA	5A	7D	78	0B	95	E3	AD	74	98	3B	36
F	64	6D	DC	F0	59	A9	4C	17	7F	91	B8	C9	57	1B	E0	61

Рисунок 3.16 – Приклад підстановки з нелінійністю 104

Крім того, за дослідженням [32] виявлено, що коли розмір кроку дуже малий, тобто коли кількість ітерацій недостатня, програма може отримати необхідне базове можливе рішення, зрештою збільшуючи кількість ітерацій достатньо, щоб зменшити негативний ефект занадто малого розміру кроку для отримання правильного рішення.

Отже, н сьогоднішній день не існує однозначного набору критеріїв ідеального S-блоку. Багато досліджень показують, що ідеальних замін, ймовірно, не існує. Однак є великі шанси досягнути бажаної нелінійності, швидкості та ефективності алгоритму при правильному відборі параметрів або при відповідних модифікаціях алгоритмів.

ВИСНОВКИ

Розробка алгоритму шифрування – непростий процес. Чимало пропозицій можна легко зламати без дотримання конкретних вказівок щодо проектування та проведення всебічного криптоаналізу. Алгоритм шифрування зображення використовує надійні компоненти сучасної криптології. Таким чином, був використаний підхід перевіреного безпечного проектування.

Однією з головних причин широкого використання електронної комерції, безсумнівно, є перевірена безпека криптографічних алгоритмів. Однак, прогрес у галузі криптоаналізу разом із технологічним розвитком постійно загрожує безпеці сучасних алгоритмів шифрування. Незважаючи на те, що сучасні алгоритми шифрування є математично безпечними, атаки на програми показали, що можуть існувати різні недоліки. Одним із криптографічних компонентів, які постраждали від атак додатків, були структури S-блоку. У криптографії S-блок – засаднича складова шифрування з симетричними ключами, яка виконує підстановки. По суті, це – звичайна таблиця підстановки. У блочних шифрах її здебільшого використовують для приховування зв’язків між ключем і шифротекстом – властивість плутанини введеної Шенноном. Загалом, S-блок приймає m біт на вхід і перетворює їх в n біт на виході, де n не завжди дорівнює m.

Нелінійність – термін, який використовується в статистиці для опису ситуації, коли немає прямого зв’язку між незалежною змінною та залежною змінною. У нелінійній залежності зміни на виході не змінюються прямо пропорційно змінам на будь-якому з вхідних даних, у той час як лінійний зв`язок створює пряму лінію на графіку, нелінійний зв’язок не створює пряму лінію, а замість цього створює криву. Безпека даних залежить від процесу

підстановки. Підстановка – нелінійне перетворення, яке здійснює заплутаність бітів. Він надає криптосистемі властивість плутанини, описану Шенноном.

Основні властивості S-блоку:

1) Надійність.
2) Збалансованість.
3) Строгий лавинний критерій.
4) Нелінійність.
5) Лінійна апроксимація.
6) Алгебраїчна складність.
7) Фіксовані (Fp) і протилежні фіксовані точки (OFp).
8) Критерій незалежності бітів.

Три основні класи методів для генерації S-боксів і булевих функцій з бажаною нелінійністю:

– Випадковий пошук
– Методи побудови
– Еволюційний (або генетичний) пошук найпростішого методу – випадковий пошук.

Математична оптимізація – це вибір відповідних елементів за деякими критеріями з багатьох можливих альтернатив. У більш загальному сенсі оптимізація включає пошук «найкращого доступного» значення цільової функції в заданій області (або вхідних даних), що містить різні типи цільових функцій і різні типи включених областей.

В роботі визначено, що евристичні алгоритми належать до трьох основних підходів до вирішення:

1) Послідовна конструкція – наприклад Dsatur , Iterated Greedy дуже швидкі методи, але не особливо ефективні;
2) Локальний пошук; пошук табу; імітація відпалу; змінний простір пошуку або пошук по змінному сусідству; повторний локальний пошук
3) Еволюційні популяційні гібридні або розподілені методи

В даній роботі було проаналізовано ряд евристичних алгоритмів пошуку S-блоків, а саме: метод градієнтного спуску, сходження на пагорб, генетичні алгоритми та алгоритм імітації відпалу. Кожен з алгоритмів є ефективним при правильно підібраних параметрах. Алгоритм сходження на пагорб зміг досягнути нелінійності в 104, однак час, який зайняла дана процедура був більшим ніж наприклад час, використаний на пошук в генетичних алгоритмах. Алгоритм імітації відпалу зміг досягнути нелінійності в 102, одна він виявився одних з найшвидших. Щодо методу градієнтного спуску, в стандартній інтерпретації даний алгоритм не зміг досягнути високої нелінійності, однак він визначився високим показником щодо швидкості та він потребує більш детальних досліджень. Генетичний алгоритм зміг досягнути нелінійності в 104 та швидкість була доволі високою, що, на мою думку, підіймає його на сходинку вище ніж інші алгоритми.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Казимиров А.В. Методы и средства генерации нелинейных узлов замены для симметричных криптоалгоритмов. (дата звернення: 11.06.2022)
2. Luke O'Connor, Andrew Klapper. Algebraic nonlinearity and its applications to cryptography. URL: http://www.cs.engr.uky.edu/~klapper/pdf/nonlinearity.pdf (дата звернення: 20.01.2022)
3. Глибовець М. М., Гуляєва Н. М. Еволюційні алгоритми: підручник. Київ: НаУКМА, 2013. С. 9-18. (дата звернення: 20.06.2022)
4. Грибков С. В., Кононова В. О., Харкянен О. В. Оцінка засобів захисту інформаційних ресурсів. *Вісник Національного університету «Львівська політехніка»*. 2014. № 806. С. 99-105. (дата звернення: 16.07.2022)
5. Гуляницький Л. Ф., Мулеса О. Ю. Прикладні методи комбінаторної оптимізації: навч. посіб. Київ: Видавничо-поліграфічний центр «Київський університет», 2016. 142 с. (дата звернення: 16.07.2022)
6. Kamsiah Mohamed, M N M Pauzi Pauzi, Fakariah Hani Hj Mohd Ali, Suriyani Ariffin. Study of S-box Properties in Block Cipher. URL: https://www.researchgate.net/publication/277613994_Study_of_S-box_Properties_in_Block_Cipher (дата звернення: 16.07.2022)
7. The Advanced Encryption Standard (AES). URL: http://www.facweb.iitkgp.ac.in/~sourav/AES.pdf (дата звернення: 17.07.2022)
8. Kazys Kazlauskas, Jaunius Kazlauskas. Key-Dependent S-Box Generation in AES Block Cipher System. URL: https://www.researchgate.net/publication/220073823_Key-Dependent_S-Box_Generation_in_AES_Block_Cipher_System (дата звернення: 17.07.2022)

9. A. F. Webster, Stafford E Tavares. On the design of S-Boxes. URL: https://www.researchgate.net/publication/2605829_On_the_design_of_S-Boxes (дата звернення: 27.07.2022)
10. Дибкова Л. М. Інформатика та комп`ютерна техніка: навч. посіб. Київ: Академвидав, 2007. 416 с. (дата звернення: 27.07.2022)
11. Випадковий пошук. URL: https://uk.wikipedia.org/wiki/%D0%92%D0%B8%D0%BF%D0%B0%D0%B4%D0%BA%D0%BE%D0%B2%D0%B8%D0%B9_%D0%BF%D0%BE%D1%88%D1%83%D0%BA (дата звернення: 8.08.2022)
12. Зайченко О. Ю., Зайченко Ю. П. Дослідження операцій. Збірник задач. Київ: Видавничий Дім «Слово», 2007. 472 с. (дата звернення: 8.08.2022)
13. Daniel Cosmin Porumbel. Heuristic Algorithms And Learning Techniques: Applications to the Graph Coloring Problem. URL: http://cedric.cnam.fr/~porumbed/papers/theseEn.pdf (с.6-7) (дата звернення: 08.08.2022)
14. Ivica Martinjak, Marin Golub. Comparison of Heuristic Algorithms in Functions Optimization and Knapsack Problem. URL: http://www.zemris.fer.hr/~golub/clanci/iis2006.pdf (дата звернення: 19.08.2022)
15. Linda Burnett. Heuristic Optimization of Boolean Functions and Substitution Boxes for Cryptography. URL: https://core.ac.uk/download/pdf/10884722.pdf (53-67) (дата звернення: 3.09.2022)
16. Local search. URL: https://docs.optaplanner.org/6.0.0.CR5/optaplanner-docs/html/localSearch.html#d0e7386 (дата звернення: 3.09.2022)
17. Richard Lathrop. Local search algorithms. URL: https://www.ics.uci.edu/~rickl/courses/cs-171/cs171-lecture-slides/2020_SS1_CS171/chap_4_Local_Search.pdf (дата звернення: 3.09.2022)

18. William Millan, Andrew John Clark. Smart Hill Climbing Finds Better Boolean Functions. URL: https://www.researchgate.net/profile/Andrew-Clark-42/publication/2357166_Smart_Hill_Climbing_Finds_Better_Boolean_Functions/links/00b4952eb3c5615876000000/Smart-Hill-Climbing-Finds-Better-Boolean-Functions.pdf (дата звернення: 3.09.2022)
19. Hill Climbing Algorithm in Artificial Intelligence. URL: https://www.javatpoint.com/hill-climbing-algorithm-in-ai (дата звернення: 20.09.2022)
20. Evolutionary Algorithms: genetic algorithms. URL: https://freecontent.manning.com/evolutionary-algorithms-genetic-algorithms/ (дата звернення: 20.09.2022)
21. Muhammad Irshad Nazeer, Ghulam Ali Mallah, Noor Ahmed Shaikh, Rakhi Bhatra, Raheel Ahmed Memon, Muhammad Ismail Mangrio. Implication of Genetic Algorithm in Cryptography to Enhance Security. URL: https://thesai.org/Downloads/Volume9No6/Paper_51-Implication_of_Genetic_Algorithm.pdf (дата звернення: 22.09.2022)
22. Graham Kendal. Simulated Annealing, URL: http://syllabus.cs.manchester.ac.uk/pgt/2017/COMP60342/lab3/Kendall-simulatedannealing.pdf (дата звернення: 22.09.2022)
23. Local Search and Optimization. URL: https://courses.cs.washington.edu/courses/csep573/11wi/lectures/04-lsearch.pdf (дата звернення: 02.10.2022)
24. Yaron Singer. Advanced Optimization. URL: https://people.seas.harvard.edu/~yaron/AM221-S16/lecture_notes/AM221_lecture9.pdf (дата звернення: 02.10.2022)
25. А.Д. Кожухівський, І.Д. Горбенко, Г.І. Гайдур, О.А. Кожухівська, В.В. Марченко. Математичні методи криптології/Навчальний посібник. URL: https://dut.edu.ua/uploads/l_2220_41037869.pdf (дата звернення: 02.10.2022)

26. ManzarSaeed, M.Saleem Mian. Methods of finding multiplicative inverses in GF(2^8). URL: https://www.sciencedirect.com/science/article/abs/pii/S0140366408004738 (дата звернення: 06.10.2022)

27. О.О. Кузнецов, М.О. Полуяненко, С.Л. Бердник, С.О. Кандій, Ю.О. Заиченко. Оптимізація параметрів алгоритму локального пошуку для генерації нелінійних підстановок. URL: http://rt.nure.ua/article/view/246433/243921 (дата звернення: 06.10.2022)

28. Nikolay Poluyanenko, Alexandr Kuznetsov, Sergey Kandiy, Anna Aryshchenko, Volodymyr Zvieriev. Simulation and Optimization of WHS Objective Function Parameters for Generating Nonlinear Substitutions (дата звернення: 11.10.2022)

29. Alexandr Kuznetsov, Lukasz Wieclaw, Nikolay Poluyanenko, Lukasz Hamera, Sergey Kandiy, Yelyzaveta Lohachova. Optimization of a Simulated Annealing Algorithm for S-Boxes Generating. URL: https://www.mdpi.com/1424-8220/22/16/6073/htm (дата звернення: 11.10.2022)

30. Kuznetsov O., Gorbenko Y., Poluyanenko M., Kandiy, S., Matveeva E. Properties of the cost function in the iterative algorithm for generating nonlinear substitution. URL: http://rt.nure.ua/article/view/262487 (дата звернення: 11.10.2022)

31. Oleksandr Kazymyrov, Valentyna Kazymyrova, Roman Oliynykov. A Method For Generation Of High-Nonlinear S-Boxes Based On Gradient Descent. URL: https://eprint.iacr.org/2013/578.pdf (дата звернення: 18.10.2022)

32. Wei Wei, Bin Zhou, Rytis Maskeliūnas, Robertas Damaševičius, Dawid Połap, Marcin Woźniak. Iterative Design and Implementation of Rapid Gradient Descent Method (дата звернення: 02.11.2022)

ДОДАТОК А

Сертифікат учасника NGSec 2022: International Conference on Next Generation Cybersecurity Systems and Applications

Рисунок А.1 – Сертифікат учасника конференції